An Introduction to Multiple Sequence Alignment — and the T-Coffee Shop.

Beyond just aligning sequences:
How good can you make your alignment, and so what?

author:

Steven M. Thompson

School of Computational Science,
Florida State University, Tallahassee, FL, 32306-4120

e-mail:

stevet@bio.fsu.edu

mailing address and phone:

2538 Winnwood Circle, Valdosta, GA, 31601-7953, 229-249-9751

Introduction

What can we know about a biological molecule, given its nucleotide or amino acid sequence? How does it fit into a particular system in some organism? What is its role in some network? We may be able to learn about it by searching for particular patterns within it that may reflect some function, such as the many motifs ascribed to catalytic activity; we can look at its overall content and composition, such as do several of the gene finding algorithms; we can map its restriction enzyme or protease cut sites; and on and on. However, what about comparisons with other sequences? Is this worthwhile? Yes, naturally it is: inference through homology is fundamental to all the biological sciences. We can learn a tremendous amount by comparing and aligning our sequence against others.

Furthermore, the power and sensitivity of sequence based computational methods dramatically increase with the addition of more data. More data yields stronger analyses — if done carefully! Otherwise, it can confound the issue. The patterns of conservation become ever clearer by comparing the conserved portions of sequences amongst a larger and larger dataset. Those areas most resistant to change are most important to the molecule, and to the system that molecule interacts with. The basic assumption is that those portions of sequence of crucial structural and functional value are most constrained against evolutionary change. They will not tolerate many mutations. Not that mutation does not occur in these regions, just that most mutation in the area is lethal, so we never see it. Other areas of sequence are able to drift more readily, being less subject to this evolutionary pressure. Therefore, sequences end up a mosaic of quickly and slowly changing regions over evolutionary time.

However, in order to learn anything by comparing sequences, we need to know how to compare them. We can use those constrained portions as 'anchors' to create a sequence alignment allowing comparison, but this brings up the alignment problem and 'similarity.' It is easy to see that sequences are aligned when they have identical symbols at identical positions, but what happens when symbols are not identical, or the sequences are not the same length. How can we know when the most similar portions of our sequences are aligned, when is an alignment optimal, and does optimal mean biologically correct?

A 'brute force,' naïve approach just won't work. Even without considering the introduction of gaps, the computation required to compare all possible alignments between just two sequences requires time proportional to the product of the lengths of the two sequences. Therefore, if two sequences are approximately the same length (N), this is a N^2 problem. The calculation would have to repeated 2N times to examine the possibility of gaps at each possible position within the sequences, now a N^{4N} problem. Waterman (1989) pointed out that using this naïve approach to align two sequences, each 300 symbols long, would require10^{88} comparisons, more than the number of elementary particles estimated to exist in the universe, and clearly impossible to solve! Part of the solution to this problem is the dynamic programming algorithm, as applied to sequence alignment, and this will be reviewed next.

Dynamic programming

Dynamic programming is a widely applied computer science technique, often used in many disciplines whenever optimal substructure solutions can provide an optimal overall solution. Let's begin with an overview of sequence alignment dynamic programming with just two sequences. I'll use an incredibly oversimplified example first: we'll consider matching symbols to be worth one point, and will not consider gapping at all. The solution occurs in two stages. The first begins very much like dot matrix methods; the second is totally different. Instead of calculating the 'score matrix' on the fly, as is often taught as one proceeds through the graph, I like to completely fill in an original 'match matrix' first, and then add points to those positions that produce favorable alignments next. I also like to illustrate the process working through the cells, many authors prefer to work through the edges; they are equivalent. Points are added based on a "looking-back-over-your-left-shoulder" algorithm rule where the only allowable trace-back is diagonally behind and above. The illustration follows below in Table 1.

Table 1. Dynamic programming without gap costs.

a) The completed match matrix using one point for matching and zero points for mismatching:

	S	C	A	T	S
A	0	0	1	0	0
C	0	1	0	0	0
T	0	0	0	1	0
S	1	0	0	0	1

b) Now begin to add points based on the best path through the matrix, always working diagonally, left to right and top to bottom. Keep track of those best paths. The second row is completed here:

	S	C	A	T	S
A	0	0	**1**	0	0
C	0	1	0	**0+1**	**0+1**
T	0	0	0	1	0
S	1	0	0	0	1

c) Continue adding points based on the best previous path through the matrix. The third row is completed here:

	S	C	A	T	S
A	0	0	**1**	0	0
C	0	**1**	0	**1**	**1**
T	0	0	**0+1**	**1+1**	**0+1**
S	1	0	0	0	1

d) The score matrix is now complete:

	S	C	A	T	S
A	0	0	**1**	0	0
C	0	**1**	0	**1**	**1**
T	0	0	**1**	**2**	**1**
S	1	0	**0+1**	**0+1**	**1+2**

e) Now pick the bottom, right-most, highest score in the matrix and work your way back through it, in the opposite direction as you arrived. This is called the trace-back stage, and the matrix is now referred to as the path graph. In this case that highest score is in the right-hand corner, but it need not be:

	S	C	A	T	S
A	0	0	**1**	0	0
C	0	**1**	0	**1**	**1**
T	0	0	**1**	**2**	**1**
S	1	0	**1**	**1**	**3**

f) Only the best tracebacks are now shown in outline characters. They are both optimal alignments:

	S	C	A	T	S
A	0	0	**1**	0	0
C	0	**1**	0	1	1
T	0	0	1	**2**	1
S	1	0	1	1	**3**

Here are the two alignments from the above path graph (f). They both have a score of three, the three matches found by the algorithm, and the highest score in the bottom row of the solved matrix:

```
SCATS        SCA.TS
 | ||          | ||
AC.TS        ..ACTS
```

Most software will arbitrarily (based on some internal rule) choose one of these to report as optimal. Some programs offer a HighRoad/LowRoad option to help explore this solution space.

The next example will be slightly more difficult. Unlike the previous example without gap penalties, I will now impose a very simple gap penalty function. Matching symbols will still be worth one point, non-matching symbols will still be worth zero points, but we will penalize the scoring scheme by subtracting one point for every gap inserted, unless they are at the beginning or end of the sequence. In other words, end gaps will not be penalized; therefore, both sequences do not have to begin or end at the same point in the alignment.

This zero penalty end-weighting scheme is the default for most alignment programs, but can often be changed with a program option, if desired. However, the linear gap function described here, and used in the example below, is a much simpler gap penalty function than normally used in alignment programs. Normally (Gotoh, 1982) an 'affine,' function is used, the standard 'y = mx + b' equation for a line that does not cross the X,Y origin, where 'b,' the Y intercept, describes how much initial penalty is imposed for creating each new gap:

$$\text{total penalty} = ([\text{length of gap}] * [\text{gap extension penalty}]) + \text{gap opening penalty}$$

To run most alignment programs with the type of simple linear DNA gap penalty used in my example below, you would have to designate a gap 'creation' or 'opening' penalty of zero, and a gap 'extension' penalty of whatever counts in that particular program as an identical base match for DNA sequences.

My example uses two random sequences that fit the TATA promoter region consensus of eukaryotes and of bacteria. The most conserved bases within the consensus are capitalized by convention. The eukaryote promoter sequence is along the X-axis, and the bacterial sequence is along the Y-axis in the following example. The solution illustration begins in the left panel in Table 2 below.

Table 2. Dynamic programming with a constant, linear gap cost.

a) First complete a match matrix using one point for matching and zero points for mismatching between bases, just like in the previous example:

	c	T	A	T	A	t	A	a	g	g
c	1	0	0	0	0	0	0	0	0	0
g	0	0	0	0	0	0	0	0	1	1
T	0	1	0	1	0	1	0	0	0	0
A	0	0	1	0	1	0	1	1	0	0
t	0	1	0	1	0	1	0	0	0	0
A	0	0	1	0	1	0	1	1	0	0
a	0	0	1	0	1	0	1	1	0	0
T	0	1	0	1	0	1	0	0	0	0

b) Now add and subtract points based on the best path through the matrix, working diagonally, left to right and top to bottom, just as before. However, when you have to jump a box to make the path, subtract one point per box jumped, except at the beginning or end of the alignment, so that end gaps are not penalized. Fill in all additions and subtractions, calculate the sums and differences as you go, and keep track of the best paths. The score matrix is shown with all calculations below:

	c	T	A	T	A	t	A	a	g	g
c	**1**	**0**	0	0	0	0	0	0	0	0
g	0	**0+1 =1**	**0+0 -0 =0**	0+0 -0 =0	0+0 -0 =0	0+0 -0 =0	0+0 -0 =0	0+0 -0 =0	**1+0 -0 =1**	1+0 =1
T	0	1+1 -1 =1	**0+1 =1**	**1+0 or+ 1-1 =1**	0+0 -0 =0	1+0 -0 =1	0+0 -0 =0	0+0 -0 =0	0+0 -0 =0	**0+1 =1**
A	0	0+0 -0 =0	**1+1 =2**	**0+1 =1**	**1+1 =2**	0+1 -1 =0	**1+1 =2**	1+1 -1 =1	0+0 -0 =0	0+0 -0 =0
t	0	1+0 -0 =1	0+1 -1 =0	**1+2 =3**	**0+1 =1**	**1+2 =3**	**0+2 -1 =1**	**0+2 =2**	**0+1 =1**	0+0 -0 =0
A	0	0+0 -0 =0	**1+1 =2**	**0+2 -1 =1**	**1+3 =4**	**0+3 -1 =2**	**1+3 =4**	**1+3 -1 =3**	**0+2 =2**	**0+1 =1**
a	0	0+0 -0 =0	1+0 -0 =1	**0+2 =2**	**1+3 -1 =3**	**0+4 =4**	**1+4 -1 =4**	**1+4 =5**	**0+3 =3**	**0+2 =2**
T	0	1+0 -0 =1	0+0 -0 =0	**1+1 =2**	**0+2 =2**	**1+3 =4**	**0+4 =4**	**0+4 =4**	**0+5 =5**	**0+5 -1 =4**

c) Clean up the score matrix next. I'll only show the totals in each cell in the matrix shown below. All best paths are highlighted:

	c	T	A	T	A	t	A	a	g	g
c	**1**	**0**	0	0	0	0	0	0	0	0
g	0	**1**	**0**	0	0	0	0	0	1	1
T	0	1	**1**	**1**	0	1	0	0	0	**1**
A	0	0	**2**	**1**	**2**	0	**2**	1	0	0
t	0	1	0	**3**	**1**	**3**	**1**	**2**	**1**	0
A	0	0	**2**	**1**	**4**	**2**	**4**	**3**	**2**	**1**
a	0	0	1	**2**	**3**	**4**	**4**	**5**	**3**	**2**
T	0	1	0	**2**	**2**	**4**	**4**	**4**	**5**	**4**

d) Finally, convert the score matrix into a trace-back path graph by picking the bottom-most, furthest right and highest scoring coordinate. Then choose the trace-back route that got you there, to connect the cells all the way back to the beginning using the same 'over-your-left-shoulder' rule. The two best trace-back routes are now highlighted with outline font in the trace-back matrix below:

	c	T	A	T	A	t	A	a	g	g
c	**1**	**0**	0	0	0	0	0	0	0	0
g	0	**1**	**0**	0	0	0	0	0	1	1
T	0	1	1	**1**	0	1	0	0	0	1
A	0	0	2	1	**2**	0	2	1	0	0
t	0	1	0	3	1	**3**	1	2	1	0
A	0	0	2	1	4	2	**4**	3	2	1
a	0	0	1	2	3	4	4	**5**	3	2
T	0	1	0	2	2	4	4	4	**5**	4

There may be more than one best path through the matrix. This time, starting at the top and working down as we did, then tracing back, I found two optimal alignments, each with a final score of 5, using our example's zero/one scoring scheme. This score is the highest, bottom-right value in the trace-back path graph, the sum of six matches minus one interior gap in one path, and the sum of five matches minus no interior gaps in the other. This score is the number optimized by the algorithm, not any type of a similarity or identity percentage! This first path is the GCG Wisconsin Package (1982-2007) Gap program HighRoad alignment found with this example's parameter settings (note that GCG uses a score of 10 for a base match here, not 1):

```
GAP of: Euk_Tata.Seq  to: Bact_Tata.Seq

Euk_Tata: A random Eukaryotic promoter TATA Box
Preferred region:  center between -36 and -20.
Bact_Tata: A random sequence that fits the consensus from the
standard E. coli RNA polymerase promoter 'Pribnow' box -10 region.

        Gap Weight:  0            Average Match: 10.000
     Length Weight: 10         Average Mismatch:  0.000

        HighRoad option              LowRoad option

         Quality:  50                 Quality:  50

            Ratio:  6.250                Ratio:  6.250
Percent Similarity: 75.000   Percent Similarity: 62.500
           Length:     10               Length:     10
             Gaps:      2                 Gaps:      0
  Percent Identity: 75.000     Percent Identity: 62.500

      1 cTATAtAagg 10              1 cTATAtAagg 10
        |  |||||                        |||||
      1 cg.TAtAaT. 8               1 .cgTAtAaT. 8
```

The GCG LowRoad alignment is my second, equivalent path. Notice that even though it has 62.5% identity as opposed to 75% identity in the HighRoad alignment, it has exactly the same score! Another way to explore dynamic programming's solution space is to reverse the entire process. This can often discover other alignments, so reverse the sequences to see alternatives. To recap, and for those people that like mathematics, an optimal pairwise alignment is defined as an arrangement of two sequences, 1 of length *i* and 2 of length *j*, such that:

1) you maximize the number of matching symbols between 1 and 2;
2) you minimize the number of gaps within 1 and 2; and
3) you minimize the number of mismatched symbols between 1 and 2.

Therefore, the actual solution can be represented by the following recursion:

$$S_{ij} = s_{ij} + \max \begin{cases} S_{i-1\ j-1} & \text{or} \\ \max S_{i-x\ j-1} + w_{x-1} & \text{or} \\ 2 < x < i & \\ \max S_{i-1\ j-y} + w_{y-1} & \\ 2 < y < i & \end{cases}$$

where Sij is the score for the alignment ending at i in sequence 1 and j in sequence 2,

sij is the score for aligning i with j,

wx is the score for making a x long gap in sequence 1,

wy is the score for making a y long gap in sequence 2,

allowing gaps to be any length in either sequence.

However, just because dynamic programming guarantees an optimal alignment, it is not necessarily the only optimal alignment. Furthermore, the optimal alignment is not necessarily the 'right' or biologically relevant alignment! Significance estimators, such as Expectation values and Monte Carlo simulations can give you some handle on this, but always question the results of any computerized solution based on what you know about the biology of the system. The above example illustrates the Needleman and Wunsch (1970) global solution. Later refinements (Smith and Waterman, 1981) demonstrated how dynamic programming could also be used to find optimal local alignments. To solve dynamic programming using local alignment (without going into all the gory details) programs use the following two tricks:

1) Mismatches are penalized by using a match function that assigns negative numbers to them. Therefore, bad paths quickly become very bad. This leads to a trace-back path matrix with many alternative paths, most of which do not extend the full length of the graph.
2) The best trace-back within the overall graph is chosen. This does not have to begin or end at the edges of the matrix — it's the best segment of alignment.

Significance

A particularly common misunderstanding regards the concept of homology versus similarity: there is a huge difference! Similarity is merely a statistical parameter that describes how much two sequences, or portions of them, are alike according to some set scoring criteria. It can be normalized to ascertain statistical significance as in database searching methods, but it's still just a number. Homology, in contrast and by definition, implies an evolutionary relationship — more than just the fact that all life evolved from the same primordial 'slime.' You need to be able to demonstrate some type of evolutionary lineage between the organisms or genes of interest in order to claim homology. Better yet, demonstrate experimental evidence, structural, morphological, genetic, or fossil, that corroborates your assertion. There really is no such thing as percent homology; something is either homologous or it's not. Walter Fitch (personal communication) explains with the joke, "homology is like pregnancy — you can't be 45% pregnant, just like something can't be 45% homologous. You either are or you are not." Do not make the mistake of calling any old sequence similarity homology. Highly significant similarity can argue for homology, not the other way around.

So, how do you tell if a similarity, in other words, an alignment discovered by some program, means anything? Is it statistically significant, is it truly homologous, and even more importantly, does it have anything to do with real biology? Many programs generate percent similarity scores; however, as seen above, these really don't mean a whole lot. Don't use percent similarities or identities to compare sequences except in the roughest way. They are

not optimized or normalized in any manner. Quality scores mean a lot more but are difficult to interpret. At least they take the length of similarity, all of the necessary gaps introduced, and the matching of symbols all into account, but quality scores are only relevant within the context of a particular comparison or search. The quality ratio is the metric optimized by dynamic programming divided by the length of the shorter sequence. As such it represents a fairer comparison metric, but it also is relative to the particular scoring matrix and gap penalties used in the procedure.

A traditional way of deciding alignment significance relies on an old statistics trick — Monte Carlo simulations. This type of significance estimation has implicit statistical problems; however, few practical alternatives exist for just comparing two sequences, and they are fast and easy. Monte Carlo randomization options in dynamic programming alignment algorithms compare an actual score, in this case the quality score of an alignment, against the distribution of scores of alignments of a randomized sequence. These options randomize your sequence at least 100 times after the initial alignment and then generate the jumbled alignment scores and a standard deviation based on their distribution. Comparing the mean of the randomized sequence alignment scores to the original score using a 'Z score' calculation can help you decide significance. An old 'rule-of-thumb' is if the actual score is much more than three standard deviations above the mean of the randomized scores, the analysis may be significant; if it is much more than five, than it probably is significant; and if it is above nine, than it definitely is significant. Many Z scores measure this distance from the mean using a simplistic Monte Carlo model assuming a normal Gaussian distribution, in spite of the fact that 'sequence-space' actually follows an 'extreme value distribution;' however, this simplistic approximation estimates significance quite well:

$$Z\ score = \frac{[\ (\ actual\ score\ ) - (\ mean\ of\ randomized\ scores\ )\ ]}{(\ standard\ deviation\ of\ randomized\ score\ distribution\ )}$$

When the two TATA sequences from the previous dynamic programming example are compared to one another using the same scoring parameters as before, but incorporating a Monte Carlo Z score calculation, their similarity is found to be not at all significant. The mean score based on 100 randomizations was 41.8 +/- a standard deviation of 7.4. Plugged into the formula: (50 – 41.8) / 7.4 = 1.11, i.e. there is no significance to the match in spite of 75% identity! Composition can make a huge difference — the similarity is merely a reflection of the relative abundance of A's and T's in the sequences!

The FastA (Pearson and Lipman, 1988, and Pearson, 1998), BLAST (Altschul, et al., 1990, and Altschul, et al., 1997), Profile (Gribskov, et al., 1987), and HMMer (Eddy, 1998) search algorithms, all use a similar approach but base their statistics on the distance of the query matches from the actual, or a simulated, extreme value distribution from the rest of the, 'insignificantly similar,' members of the database being searched. For alignments without gaps, the math generalizes such that the Expectation value E relates to a particular score S through the function $E = Kmne^{-\lambda S}$ (Karlin and Altschul, 1990, and see http://www.ncbi.nlm.nih.gov/BLAST/tutorial/Altschul-1.html). In a database search m is the length of the query and n is the size of the database in residues. K and λ are supplied by statistical theory, dependent on the scoring system and the background amino acid frequencies, and calculated from actual or simulated database alignment distributions. Expectation values are printed in

scientific notation and the smaller the number, i.e. the closer it is to 0, the more significant the match. Expectation values show us how often we should expect a particular alignment to occur merely by chance alone in a search of that size database. In other words, in order to assess whether a given alignment constitutes evidence for homology, it helps to know how strong an alignment can be expected from chance alone. Rough, conservative guidelines to Z scores and Expectation values from a typical protein search follow in Table 3.

Table 3. Rough, conservative guidelines to Z scores and Expectation values from a typical protein search.

~Z score	~E value	Inference
≤3	≥0.1	little, if any, evidence for homology, but impossible to disprove!
≅5	$\cong 10^{-2}$	probably homologous, but may be due to convergent evolution
≥10	$\leq 10^{-3}$	definitely homologous

Be very careful with any guidelines such as these, though, because they are entirely dependent on both the size and content of the database being searched as well as how often you perform the search! Think about it: the odds are way different for rolling dice depending on how many dice you roll, whether they are 'loaded' or not, and how often you try.

Another very powerful empirical method of determining significance is to repeat a database search with the entry in question. If that entry finds more significant 'hits' with the same sorts of sequences as the original search, then the entry in question is undoubtedly homologous to the original entry. That is, homology is transitive. If it finds entirely different types of sequences, then it probably is not. Modular proteins with distinctly separate domains confuse issues considerably, but the principles remain the same, and can be explained through domain swapping and other examples of non-vertical transmission. And, finally, the 'gold-standard' of homology is shared structural folds — if you can demonstrate that two proteins have the same structural fold, then, regardless of similarity, at least that particular domain is homologous between the two.

Scoring matrices

However, what about protein sequences — conservative replacements and similarities, as opposed to identities? This is certainly an additional complication. Particular amino acids are very much alike, structurally, chemically, and genetically. How can we take advantage of amino acid similarity of in our alignments? People have been struggling with this problem since the late 1960's. Dayhoff (Schwartz and Dayhoff, 1979) unambiguously aligned closely related protein datasets (no more than 15% difference, and in particular cytochrome c) available at that point in time and noticed that certain residues, if they mutate at all, are prone to change into certain other residues. As it works out, these propensities for change fell into the same categories that chemists had known for years — those same chemical and structural classes mentioned above — conserved through the evolutionary constraints of natural selection. Dayhoff's empirical observation quantified these changes. Based on the multiple sequence alignments that she created and the empirical amino acid frequencies within those alignments, the assumption that estimated mutation rates in closely related proteins can be extrapolated to more distant relationships, and matrix and logarithmic mathematics, she was able to empirically specify the relative

probabilities at which different residues mutated into other residues through evolutionary history, as appropriate within some level of divergence between the sequences considered. This is the basis of the famous PAM (corrupted acronym of 'accepted point mutation') 250 (meaning that the matrix has been multiplied by itself 250 times) log odds matrix.

Since Dayhoff's time other biomathematicians (esp. see Henikoff and Henikoff's [1992] BLOSUM series of matrices, and for a somewhat controversial matrix see Gonnet et al. [1992]) have created matrices regarded more accurate than Dayhoff's original, but the concept remains the same. Furthermore, Dayhoff's original PAM 250 matrix remains a classic as historically the most widely used amino acid substitution matrix. Collectively these types of matrices are known as symbol comparison tables, log odds matrices, and substitution or scoring matrices, and they are fundamental to all sequence comparison techniques.

The default amino acid scoring matrix for most protein similarity comparison programs is now the BLOSUM62 table (Henikoff and Henikoff, 1992). The "62" refers to the minimum level of identity within the ungapped sequence blocks that went into the creation of the matrix. Lower BLOSUM numbers are more appropriate for more divergent datasets. The BLOSUM62 matrix follows below in Table 4; values whose magnitude is ≥ ±4 are drawn in shadowed characters to make them easier to recognize.

Table 4. The BLOSUM62 amino acid scoring matrix.

	A	B	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	X	Y	Z
A	4	-2	0	-2	-1	-2	0	-2	-1	-1	-1	-1	-2	-1	-1	-1	1	0	0	-3	-1	-2	-1
B	-2	6	-3	6	2	-3	-1	-1	-3	-1	-4	-3	1	-1	0	-2	0	-1	-3	-4	-1	-3	2
C	0	-3	9	-3	-4	-2	-3	-3	-1	-3	-1	-1	-3	-3	-3	-3	-1	-1	-1	-2	-1	-2	-4
D	-2	6	-3	6	2	-3	-1	-1	-3	-1	-4	-3	1	-1	0	-2	0	-1	-3	-4	-1	-3	2
E	-1	2	-4	2	5	-3	-2	0	-3	1	-3	-2	0	-1	2	0	0	-1	-2	-3	-1	-2	5
F	-2	-3	-2	-3	-3	6	-3	-1	0	-3	0	0	-3	-4	-3	-3	-2	-2	-1	1	-1	3	-3
G	0	-1	-3	-1	-2	-3	6	-2	-4	-2	-4	-3	0	-2	-2	-2	0	-2	-3	-2	-1	-3	-2
H	-2	-1	-3	-1	0	-1	-2	8	-3	-1	-3	-2	1	-2	0	0	-1	-2	-3	-2	-1	2	0
I	-1	-3	-1	-3	-3	0	-4	-3	4	-3	2	1	-3	-3	-3	-3	-2	-1	3	-3	-1	-1	-3
K	-1	-1	-3	-1	1	-3	-2	-1	-3	5	-2	-1	0	-1	1	2	0	-1	-2	-3	-1	-2	1
L	-1	-4	-1	-4	-3	0	-4	-3	2	-2	4	2	-3	-3	-2	-2	-2	-1	1	-2	-1	-1	-3
M	-1	-3	-1	-3	-2	0	-3	-2	1	-1	2	5	-2	-2	0	-1	-1	-1	1	-1	-1	-1	-2
N	-2	1	-3	1	0	-3	0	1	-3	0	-3	-2	6	-2	0	0	1	0	-3	-4	-1	-2	0
P	-1	-1	-3	-1	-1	-4	-2	-2	-3	-1	-3	-2	-2	7	-1	-2	-1	-1	-2	-4	-1	-3	-1
Q	-1	0	-3	0	2	-3	-2	0	-3	1	-2	0	0	-1	5	1	0	-1	-2	-2	-1	-1	2
R	-1	-2	-3	-2	0	-3	-2	0	-3	2	-2	-1	0	-2	1	5	-1	-1	-3	-3	-1	-2	0
S	1	0	-1	0	0	-2	0	-1	-2	0	-2	-1	1	-1	0	-1	4	1	-2	-3	-1	-2	0
T	0	-1	-1	-1	-1	-2	-2	-2	-1	-1	-1	-1	0	-1	-1	-1	1	5	0	-2	-1	-2	-1
V	0	-3	-1	-3	-2	-1	-3	-3	3	-2	1	1	-3	-2	-2	-3	-2	0	4	-3	-1	-1	-2
W	-3	-4	-2	-4	-3	1	-2	-2	-3	-3	-2	-1	-4	-4	-2	-3	-3	-2	-3	11	-1	2	-3
X	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Y	-2	-3	-2	-3	-2	3	-3	2	-1	-2	-1	-1	-2	-3	-1	-2	-2	-2	-1	2	-1	7	-2
Z	-1	2	-4	2	5	-3	-2	0	-3	1	-3	-2	0	-1	2	0	0	-1	-2	-3	-1	-2	5

Notice that positive identity values range from 4 to 11, and negative values for rare substitutions go as low as -4. The most conserved residue is tryptophan with an identity score of 11; cysteine is next with a score of 9; histidine

gets 8; both proline and tyrosine get scores of 7. These residues get the highest scores because of two biological factors: they are very important to the structure and function of proteins, and they are the rarest amino acids found in nature. Also check out the hydrophobic substitution triumvirate — isoleucine, leucine, valine, and to a lesser extent methionine — all easily swap places. So, rather than using the zero/one match function that we used in the previous dynamic programming examples, protein sequence alignments use the match function provided by an amino acid scoring matrix. The concept of similarity becomes very important with some amino acids being way 'more similar' than others!

Multiple sequence dynamic programming

Dynamic programming reduces the pairwise alignment problem's complexity to order N^2. But how do you work with more than just two sequences at a time? It becomes a much harder problem. You could manually align your sequence data with an editor, but some type of an automated solution is desirable, at least as a starting point to manual alignment. However, solving the dynamic programming algorithm for more than just two sequences rapidly becomes intractable. Dynamic programming's complexity, and hence its computational requirements, increases exponentially with the number of sequences in the dataset being compared (complexity=[sequence length]$^{\text{number of sequences}}$). Mathematically this is an N-dimensional matrix, quite complex indeed. Pairwise dynamic programming solves a two-dimensional matrix, and the complexity of the solution is equal to the length of the longest sequence squared. Well, a three-member standard dynamic programming sequence comparison would be a matrix with three axes, the length of the longest sequence cubed, and so forth. You can at least draw a three-dimensional matrix, but more than that becomes impossible to even visualize. It quickly boggles the mind!

Several different heuristics have been employed over the years to simplify the complexity of the problem. One program, MSA (Gupta et al. 1995), attempts to simultaneously solve the N-dimensional matrix recursion using a bounding box trick. However, the algorithm's complexity precludes its use in most situations, except with very small datasets. One way to simultaneously solve the algorithm and yet reduce its complexity is to restrict the search space to only the most conserved 'local' portions of all the sequences involved. This approach is used by the program PIMA (Smith and Smith, 1995). MSA and PIMA are both available through the Internet at several bioinformatics servers (in particular see the Baylor College of Medicine's Search Launcher at http://searchlauncher.bcm.tmc.edu/), or they can be installed on your own machine.

How the algorithms work

Most implementations of automated multiple alignment modify dynamic programming by establishing a pairwise order in which to build the alignment. This heuristic modification is known as pairwise, progressive dynamic programming. Originally attributed to Feng and Doolittle (1987), this variation of the dynamic programming algorithm generates a global alignment, but restricts its search space at any one time to a local neighborhood of the full length of only two sequences. Consider a group of sequences. First all are compared to each other, pairwise, using some quick variation of standard dynamic programming. This establishes an order for the set, most to least similar, a 'guide-tree' if you will. Subgroups are clustered together similarly. The algorithm then

takes the top two, most similar sequences, and aligns them. Then it creates a quasi-consensus of those two and aligns that to the third sequence. Next create the same sort of quasi-consensus of the first three sequences and align that to the forth most similar. The way that the program makes and uses this 'consensus' sequence is one of the big differences between the various implementations. This process, all using standard, pairwise dynamic programming, continues until it has worked its way through all of the sequences and/or sets of clusters, to complete the full multiple sequence alignment.

The pairwise, progressive solution is implemented in several programs. Perhaps the most popular is Higgins' and Thompson's ClustalW (1994) and its multi-platform, graphical user interface ClustalX (Thompson, et al., 1997). This program made the first major advances over the basic Feng and Doolittle algorithm by incorporating variable sequence weighting, dynamically varying gap penalties and substitution matrices, and a neighbor-joining (NJ, Saitou and Nei, 1987) guide-tree. The programs can be downloaded from the ClustalX homesite, ftp://ftp-igbmc.u-strasbg.fr/pub/ClustalX/, to install on your own machine, or they can be run through the World Wide Web (WWW) at several sites. ClustalX is available for most windowing Operating Systems: most UNIX flavors, Microsoft Windows, and Macintosh. Complete documentation comes with the program and is accessed through a "Help" menu. The GCG program PileUp implements a similar method, but without the later innovations, and ClustalW is now included in the GCG package as well.

Several more variations on the theme have come along in recent years. T-Coffee (Tree-based Consistency Objective Function For alignment Evaluation [Notredame, et al., 2000]) was one of the first after ClustalW, and it has gained much favor. It will be presented in further detail in this chapter's final section. Its biggest innovation is the use of a preprocessed, weighted library of all the pairwise global alignments between the sequences in your dataset plus the ten best local alignments associated with each pair of sequences. This helps build the NJ guide-tree and the progressive alignment both. Furthermore, the library is used to assure consistency and help prevent errors, by allowing 'forward-thinking' to see whether the overall alignment will be better one way or another after particular segments are aligned one way or another. Notredame (2006) makes the apt analogy of school schedules: everybody, students, teachers and administrators, with some folk being more important than others, i.e. the weighting factor, puts the schedule they desire in a big pile, i.e. T-Coffee's library, with the trick being to best fit all the schedules to one academic calendar, so that everybody is happiest, i.e. T-Coffee's final multiple sequence alignment. T-Coffee is one of the most accurate multiple sequence alignment methods available because of this rationale, but it is not the fastest.

Muscle (Edgar, 2004) is another relatively new multiple sequence alignment program. It is incredibly fast, yet nearly as accurate as T-Coffee. Muscle is an iterative method that uses weighted log-expectation profile scoring along with a slew of optimizations. It proceeds in three stages: draft progressive using k-mer counting, improved progressive using a revised tree from the previous iteration, and refinement by sequential deletion of each tree edge with subsequent profile realignment. Another fairly new program, MAFFT (Katoh, et al., 2005), can be run in either fast, approximate mode, using a Fast Fourier Transformation, where its capability to handle large datasets and its speed is similar to Muscle; or in a slow, iteratively refined, optimized mode, where its results and

capabilities are similar to T-Coffee. Perhaps the most accurate new multiple sequence alignment program is ProbCons (Do, et al., 2005). It uses Hidden Markov Model (HMM) techniques and posterior probability matrices that compare random pairwise alignments to expected pairwise alignments. Probability consistency transformation is used to reestimate the scores, and a guide-tree is then constructed, which is used to compute the alignment, which is then iteratively refined. These methods and more are all tied into T-Coffee as external modules, as long as they are all installed on your system.

Coding DNA issues

All of these alignment algorithms, pairwise, multiple, and database similarity searching, are far more sensitive at the amino acid level than at the DNA level. Twenty match symbols are just much easier to align then only four; the signal to noise ratio is so much better. And, the concept of similarity applies to amino acids, but generally not to nucleotides. Furthermore, many DNA base changes (especially third position changes) do not change the encoded protein. All of these factors drastically increase the 'noise' level of DNA; typically giving protein searches a much greater 'look-back' time, at least doubling it. Therefore, database searching and sequence alignment should always be done on a protein level, unless you are dealing with noncoding DNA, or if the sequences are so similar as to not cause any problems. Therefore, usually, if dealing with coding sequences, translate the DNA to its protein counterpart, before performing multiple sequence alignment.

Even if you are dealing with very similar coding sequences, where the DNA can be directly aligned, it is often best to align the DNA along with its corresponding proteins. In addition to the much more easily achieved alignment, this also insures that alignment gaps are not placed within codons. Phylogenetic analysis can then be performed on the DNA rather than on the proteins. This is especially important when dealing with datasets that are quite similar, since the proteins may not reflect many differences hidden in the DNA. Furthermore, many people prefer to run phylogenetic analyses on DNA rather than protein regardless of how similar they are — the multiple substitution models have a long and well-accepted history, and yet are far simpler. In fact, some phylogenetic inference algorithms do not even take advantage of amino acid similarity when dealing with protein sequences; they only count identities, though many others can use PAM style models. However, the more diverged a dataset becomes, the more random third and eventually first codon positions become, which introduces noise (error) into the analysis. Therefore, often third positions and sometimes first positions are masked out of datasets. Just like in most of computational molecular biology, one is always balancing signal against noise. Too much noise or too little signal, both degrade the analysis to the point of nonsense.

Several scripts and programs, as well as some Web servers, can perform this sort of codon-based alignment, but they can be a bit tricky to run. Examples include mrtrans (Pearson, 1990) (also available in EMBOSS [Rice, et al., 2000] as tranalign and in BioPerl [Stajich et al., 2000] as aa_to_dna_aln), transAlign (Bininda-Emonds, 2005), RevTrans (Wernersson and Pedersen, 2003), protal2dna (Letondal and Schuerer, [Pasteur Institute and in BioPerl]), and PAL2NAL (Suyama, et al., 2006). Dedicated sequence analysis editors, such as GCG's SeqLab

(based on Smith's Genetic Data Environment [GDE], 1994), can also be used for this in a manual process. The logic to this manual paired protein and DNA codon alignment approach follows:

1) The easy case is where you can align the DNA directly. If the DNA sequences are directly alignable because they are quite similar, then use whatever automated tool you want to create your DNA alignment and load it into your multiple sequence editor. Next use your editor's align translation function to create aligned corresponding protein sequences. Select the region to translate based on the CDS reference in each DNA sequence's annotation. Be careful of CDS entries that do not begin at position 1 — the GenBank CDS feature annotation "/codon_start=" identifies which position the translation begins within the first codon listed for each gene. You may also have to trim sequences down to just the relevant gene or exons, especially if they're genomic. Group each protein to its corresponding DNA sequence, if the option is available, so that subsequent manipulations will keep them together.

2) The way more difficult case is where you need to use the protein sequences to create the alignment because the DNA is not directly alignable. In this case you need to create the protein sequence alignment first, and then load their corresponding DNA sequences into the editor. You can find the DNA sequence accession codes in the annotation of the protein sequence entries. Next translate the unaligned DNA sequences into new protein sequences with the align translation function and group these to their corresponding DNA sequences, just as above. However, this time the DNA along with their translated sequences are not aligned as a set, just the other protein set is aligned. Also, group all of the aligned protein dataset together, separately from the DNA/aligned translation set. Now comes the manual part; slide the original aligned protein sequence set apart to match the codons of the DNA along with its aligned translation, inserting gaps in whichever set need them to recreate the alignment. Merge the newly aligned sequences into the existing alignment group as you go, and then start on the next one. It sounds difficult, but since you're matching up two identical protein sequences, the DNA translation and the original aligned protein, it's really not too bad.

Multiple sequence alignment is much more difficult if you are forced to align nucleotides because the region does not code for a protein. Automated methods may be able to help as a starting point, but they are certainly not guaranteed to come up with a biologically correct alignment. The resulting alignment will probably have to be extensively edited, if it works at all. Success will largely depend on the similarity of the nucleotide dataset.

Reliability?

One liability of most global progressive, pairwise methods is they are entirely dependent on the order in which the sequences are aligned. Fortunately ordering them from most similar to least similar usually makes biological sense and works quite well. However, the techniques are very sensitive to the substitution matrix and gap penalties specified. Some programs allow 'fine-tuning' areas of an alignment by realignment with different scoring matrices and/or gap penalties; this can be extremely helpful. However, any automated multiple sequence alignment program should be thought of as only a tool to offer a starting alignment that can be improved upon, not

the ‘end-all-to-meet-all’ solution, guaranteed to provide the ‘one-true’ answer. Although, in this post-genomics era, when having to deal with Giga bases of data, it does make sense to start with the ‘best’ solution possible. This is the premise of using a very accurate multiple sequence alignment package, such as T-Coffee (Notredame, et al., 2000).

Regardless of the program used to create an alignment, always use comparative approaches to help assure its reliability. After the program has offered its best guess, try to improve it further. Think about it — a sequence alignment is a statement of positional homology — it is a hypothesis of evolutionary history. It establishes the explicit homologous correspondence of each individual sequence position, each column in the alignment. Therefore, insure that you have prepared a good one — be sure that it makes sense — devote considerable time and energy toward developing the best alignment possible.

Editing alignments to insure that all columns are truly homologous should be encouraged. Dedicated sequence alignment editing software such as GCG’s SeqLab (1982–2007), Jalview (Clamp, et al., 2004), Se-Al (Rambaut, 1996), and SeaView (Galtier, et al., 1996) are great for this, but any editor will do, as long as the sequences end up properly formatted afterwards. Use your understanding of the system to help guide your judgment. Look for conserved functional sites and other motifs — they should all line up. Searches of the *PROSITE Database of protein families and domains* (Bairoch, 1992) for catalogued structural, regulatory, and enzymatic consensus patterns or ‘signatures’ in your dataset can help, as can *de novo* motif discovery tools like the MEME (Bailey and Elkan, 1994), MotifSearch (Bailey and Gribskov, 1998) program pair.

Make subjective decisions. Is it good enough; do things line up the way that they should? Assure that known enzymatic, regulatory, and structural elements all align. Look for columns of strongly conserved residues such as tryptophans, cysteines, and histidines; important structural amino acids such as prolines, tyrosines and phenylanines; and those conserved isoleucine, leucine, valine substitutions. If, after all else, you decide that you just can’t align some region, or an entire sequence, then get rid of it. Another alternative is to use the mask function available in some programs. Cutting an entire sequence out of an alignment may leave columns of gaps across the entire alignment that will need to be removed. The extreme amino- and carboxy-termini (5’ and 3’ in DNA) seldom align nicely; they are often jagged and uncertain, and should probably be excluded. The results of subsequent analyses are absolutely dependent upon the quality of your alignment assessment.

Researchers have successfully used the conservation of co-varying sites in ribosomal and other structural RNA alignments to assist in alignment refinement. That is, as one base in a stem structure changes the corresponding Watson-Crick paired base will change in a corresponding manner. This principle has guided the assembly of rRNA structural alignments at the Ribosomal Database Project at Michigan State University (Cole, et al. 2007, http://rdp.cme.msu.edu/) and at the University of Gent, Belgium, at the European Ribosomal RNA database (Wuyts, et al. 2004, http://www.psb.ugent.be/rRNA/).

Be sure an alignment makes biological sense — align things that make sense to align! Beware of comparing ‘apples and oranges.’ Be particularly suspect of sequence datasets found through text-based database searches

such as Entrez (NCBI) and GCG's LookUp (based on the Sequence Retrieval System [SRS] of Etzold and Argos [1993]). For example, don't try to align receptors and/or activators with their namesake proteins. Be wary of trying to align genomic sequences with cDNA when working with DNA; the introns will cause all sorts of headaches. Similarly, aligning mature and precursor proteins, or alternate splicing forms, from the same organism and locus, doesn't make evolutionary sense, as one is not evolved from the other, rather one is the other. Watch for redundant sequences; there are tons of them in the databases. If creating alignments for phylogenetic inference, either make paralogous comparisons (i.e. evolution via gene duplication) to ascertain gene phylogenies within one organism, or orthologous (within one ancestral loci) comparisons to ascertain gene phylogenies between organisms (which should imply organismal phylogenies). Try not to mix them up without complete data representation. Otherwise, confusion can mislead interpretation, especially if the sequences' nomenclature is inconsistent. These are all easy mistakes to make; try your best to avoid them.

Remember the old adage "garbage in — garbage out!" Some general guidelines to remember (Olsen, 1992) include the following:

- If the homology of a region is in doubt, then throw it out.
- Avoid the most diverged parts of molecules; they are the greatest source of systematic error.
- Do not include sequences that are more diverged than necessary for the analysis at hand.

Complications

Sequence data format is a huge problem in computational molecular biology. The major databases all have their own distinct format, plus many of the different programs and packages require their own. Clustal (Higgins, et al., 1992) has a specific format associated with it. The FastA database similarity-searching package (Pearson and Lipman, 1988) uses a very basic sequence format. The National Center for Biotechnology Information (NCBI) uses a library standard called ASN.1 (Abstract Syntax Notation One), plus it provides GenBank flatfile format for all sequence data. GCG uses three sequence formats: Single Sequence Format (SSF), Multiple Sequence Format (MSF), and SeqLab's Rich Sequence Format (RSF) that contains both sequence data and annotation. PAUP* (Phylogenetic Analysis Using Parsimony [and other methods, pronounced "pop star"] Swofford, 1989-2007), MrBayes (Ronquist and Huelsenbeck, 2003), and many other phylogenetic analysis packages, have a required format called the NEXUS file. The PAUP* interface in the GCG Package, PAUPSearch, generates NEXUS format directly from GCG alignments. Even PHYLIP (PHYLogeny Inference Package, Felsenstein, 1980-2007) has its own unique input data format. Format standards have been argued over for years, such as using XML for everything, but until everybody agrees, which is not likely to happen, it just won't happen. Fortunately several freeware programs are available to convert formats back and forth between the required standards, but it all can get quite confusing. BioPerl's SeqIO system (Stajich, 2002) and ReadSeq (Gilbert, 1990) are two of the best. T-Coffee (Notredame, et al., 2000) comes with one built in named "seq_reformat."

Alignment gaps are another problem. Different program suites may use different symbols to represent them. Most programs use hyphens, "-"; the GCG Package uses periods, ".", for interior gaps, and tildes, "~", for

placeholder gaps. Furthermore, not all gaps in sequences should be interpreted as deletions. Interior gaps are probably okay to represent this way, as regardless of whether a deletion, insertion or a duplication event created the gap, logically they are treated the same by the algorithm. These are known as 'indels.' However, end gaps should not be represented as indels, because a lack of information before or beyond the length of any given sequence may not be due to a deletion or insertion event. It may have nothing to do with the particular stretch being analyzed at all. It just may not have been sequenced! These gaps are just placeholders for the sequence. Therefore, it is safest to manually edit an alignment to change leading and trailing gap symbols to "x"'s which mean "unknown amino acid," or "n"'s which mean "unknown base," or "?"'s which is supported by many programs, but not all, and means "unknown residue or indel." This will assure that the programs don't make incorrect assumptions about your sequences.

Applicability?

Now that we understand some of the principles and problems of multiple sequence alignment, what's so great about doing it anyway; why would anyone want to bother? Multiple sequence alignments are:

- very useful in the development of PCR primers and hybridization probes;
- great for producing annotated, publication quality, graphics and illustrations;
- invaluable in structure/function studies through homology inference;
- essential for building HMM profiles for remote homology similarity searching and alignment; and
- required for molecular evolutionary phylogenetic inference programs, e.g. PAUP*, MrBayes, and PHYLIP.

A multiple sequence alignment is useful for probe and primer design by allowing you to visualize the most conserved regions of an alignment. This technique is invaluable for designing phylogenetic specific probes as it clearly localizes areas of high conservation and high variability in an alignment. Depending on the dataset that you analyze, any level of phylogenetic specificity can be achieved. Pick areas of high variability in the overall dataset that correspond to areas of high conversation in phylogenetic category subset datasets to differentiate between universal and phylo-specific potential probe sequences. After localizing general target areas on the sequence, you can then use any of several primer discovery programs, such as GCG's Prime, or MIT's Primer3 (Rozen and Skaletsky, 2000), or the commercial Oligo program (National Biosciences, Inc.), to find the best primers within those regions, and to test those potential probes for common PCR conditions and problems. See my workshop tutorial illustrating this technique using GCG and SeqLab at http://bio.fsu.edu/~stevet/PrimerDesign.pdf if you are interested. The technique is illustrated in Figure 1 below where I identify potential primer locations that should differentiate between the major capsid protein genes (L1) of the highly carcinogenic Human Papillomavirus (HPV) Type 16 strains from the rest of the Type 16 relatives.

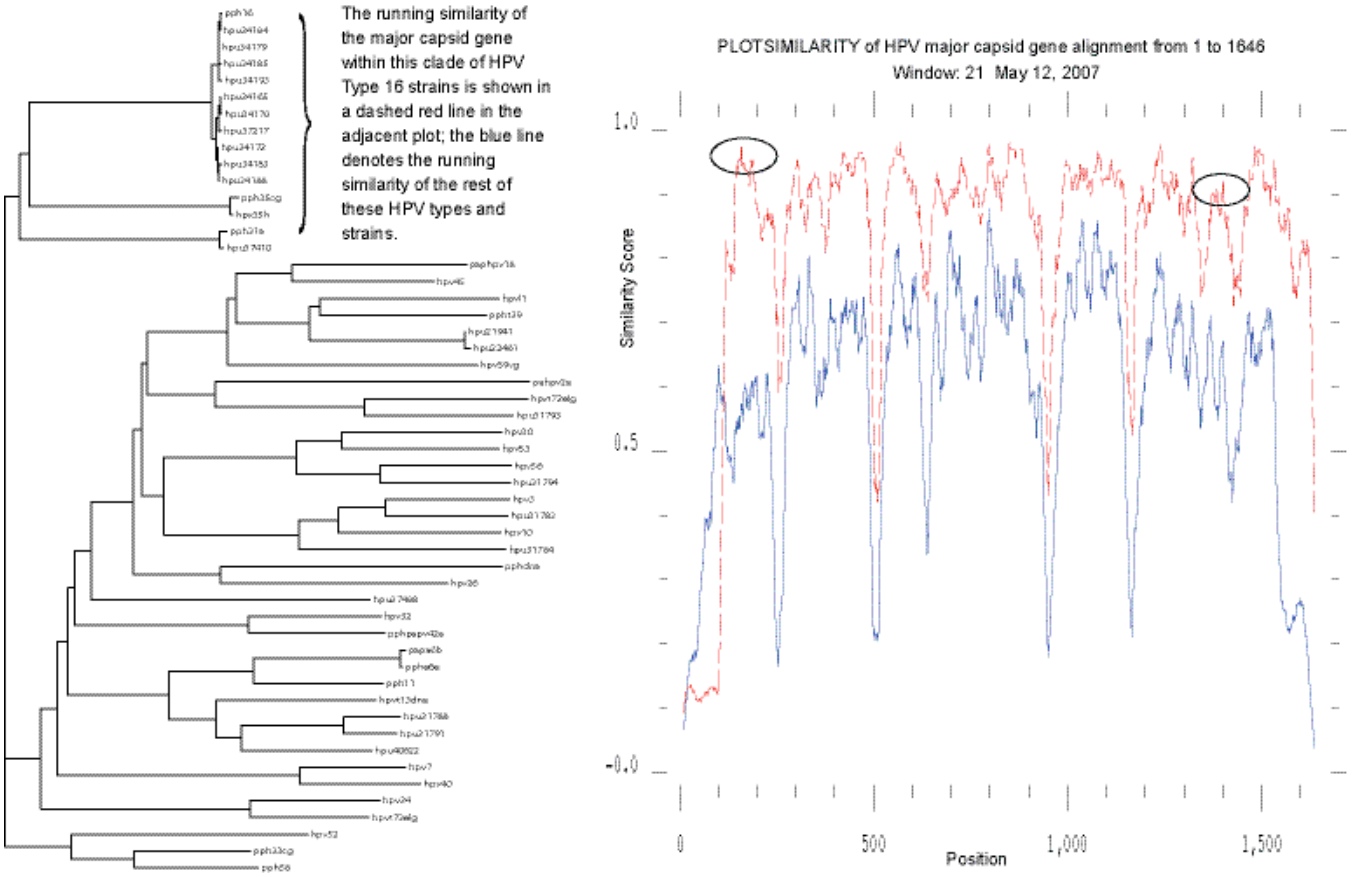

Figure 1. A phylogram of the HPV type assemblage most closely related to Type16 based on the L1 major capsid protein, and the corresponding GCG PlotSimlarity traces. The ellipses denote potential areas in which to localize PCR primers within the gene that would differentiate the Type 16 clade from it's closest relatives. These are areas of high L1 conservation in the Type 16 clade (the red, dashed line) that correspond to areas of much weaker conservation in the other clades (the blue, solid line).

Graphics prepared from multiple sequence alignments can dramatically illustrate functional and structural conservation. These can take many forms of all or portions of an alignment — shaded or colored boxes or letters for each residue (e.g. BoxShade by Hofmann and Baron at EMBnet.org, and the various PostScript output options in GCG's SeqLab), cartoon representations (e.g. WebLogos [Schneider and Stephens, 19900] and GCG's SeqLab graphical feature representation), running line graphs of overall similarity (as seen above with GCG's PlotSimilarity and as displayed by ClustalX), overlays of attributes, various consensus representations, etc. — all can be printed with high-resolution equipment, usually in color or gray tones. These can make a big difference in a poster or manuscript presentation. Figure 2 shows a multiple sequence alignment of the most conserved portion of the HMG DNA-binding domain from several paralogous members of the human HMG-box superfamily.

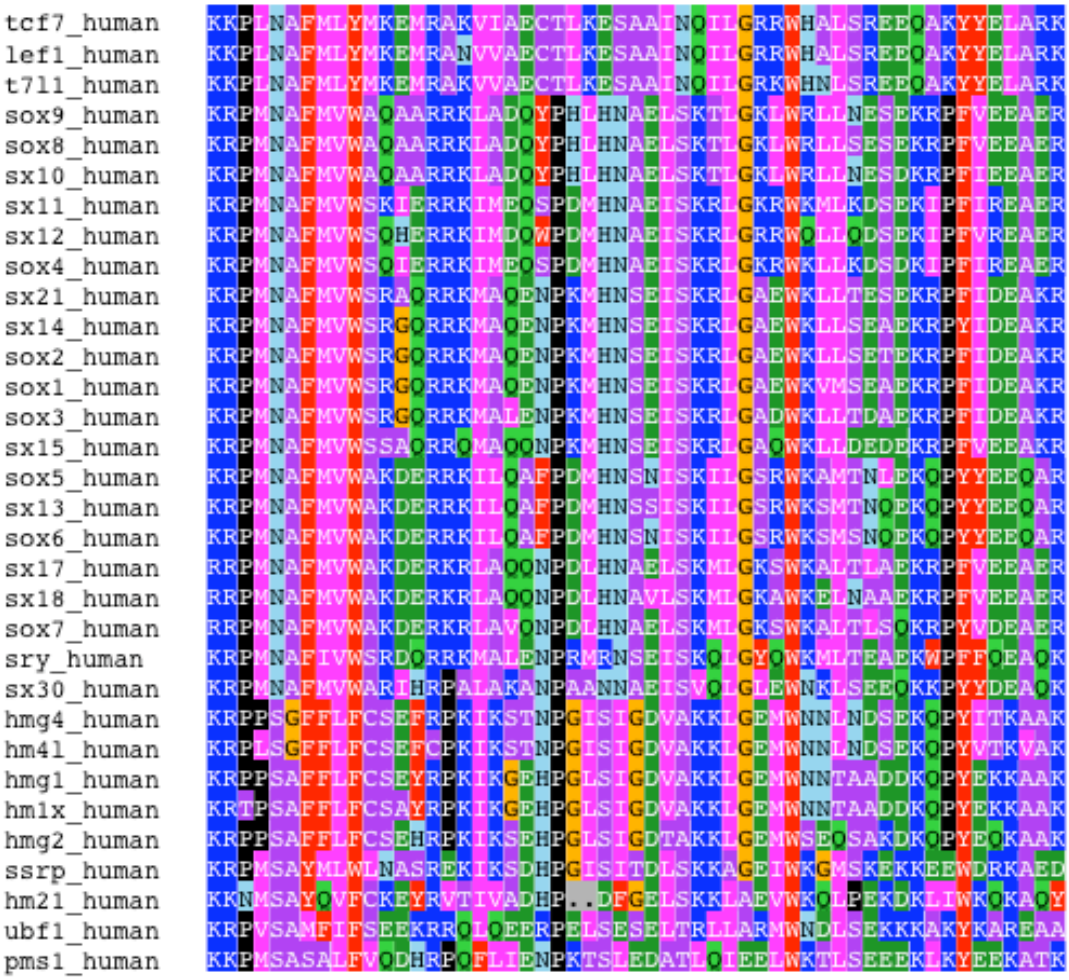

Figure 2. A GCG SeqLab PostScript graphic of the most conserved portion of the HMG-box DNA binding domain from a collection of paralogous human HMG-box protein sequences.

Conserved regions of an alignment are important. In addition to the conservation of primary sequence, structure and function is also conserved in these crucial regions. In fact, recognizable structural conservation between true homologues extends way beyond statistically significant sequence similarity. An oft-cited example is in the serine protease superfamily. *S. griseus* protease A demonstrates remarkably little sequence similarity when compared to the rest of the superfamily (Expectation values E()≥$10^{1.8}$ in a typical search) yet its three-dimensional structure clearly shows its allegiance to the serine proteases (RMSD of less than 3 Å with most of the family) (Pearson, W.R., personal communication). These principles are the premise of 'homology modeling' and it works remarkably well. An automated homology modeling tool is even available on the ExPASy server in Switzerland. Supported by the Swiss Institute of Bioinformatics (SIB) and GlaxoSmithKline, Swiss-Model (http://swissmodel.expasy.org//SWISS-MODEL.html, see Guex, et al. [1999]) has dramatically changed the homology modeling process. It is a relatively painless way to get a theoretical model of a protein structure. While not always successful, the minimal amount of effort involved in making the attempt makes it an excellent time investment. It won't always generate a homology model for your sequence, depending on how similar the closest

sequence with an experimentally solved structure is to it; however, it is a very reasonable first approach and will often lead to remarkably accurate representations. I submitted a *Giardia lamblia* Elongation Factor 1α sequence to Swiss-Model in "First Approach mode." The results were e-mailed back to me in less than five minutes. Figure 3 displays a RasMac (http://www.umass.edu/microbio/rasmol/ [see e.g. Sayle and Milner-White, 1995]) "Strands" graphic of the *Giardia* EF-1α structural model from Swiss-Model superimposed over the eight most similar solved structural templates.

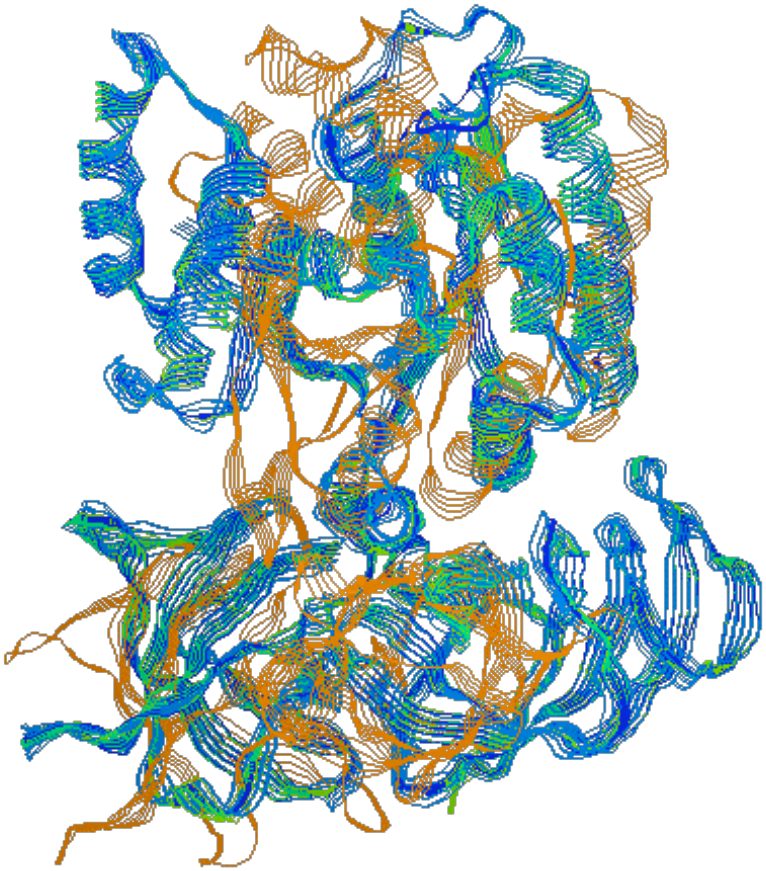

Figure 3. A RasMac representation of the Swiss-Model *Giardia* EF-1α structure superimposed over the eight most similar solved structures.

Profiles are a position specific scoring matrix (PSSM) description of an alignment or a portion of an alignment. Gap insertion is penalized more heavily in conserved areas of the alignment than it is in variable regions, and the more highly conserved a residue is, the more important it becomes. Profiles are created from an existing alignment of related sequences, and then they are used to search for remote sequence similarities and/or to build larger multiple sequence alignments. Originally described by Gribskov (1987), and then automated by NCBI's PSI-BLAST (Altschul, et al., 1997), later refinements have added more statistical rigor (see e.g. Eddy's Hidden Markov Model profiles [1996 and 1998]). The original Gribskov style profiles require a lot of time and skill to prepare and validate, and they are heuristics based. An excess of subjectivity, and a lack of formal statistical rigor also contribute as drawbacks. Eddy's HMMer (pronounced "hammer") package uses Hidden Markov modeling,

with a formal probabilistic basis and consistent gap insertion theory, to overcome these limitations. The HMMer package can build and manipulate HMMer profiles and profile databases, search sequences against HMMer profile databases and visa versa, and easily create multiple sequence alignments using HMMer profiles as a 'seed.' This ability to easily create larger and larger multiple sequence alignments is incredibly powerful and way faster than starting all over each time you want to add another sequence to an alignment. The 'take-home' message is HMMer profiles are much easier to build than traditional profiles, and they do not need to have nearly as many sequences in their alignments in order to be effective. Furthermore, they offer a statistical rigor not available in Gribskov profiles, plus they have all the sensitivity of any profile technique. In effect, they are like the 'old-fashioned' profiles pumped up on steroids! One big difference between HMMer profiles and others is when the profile is built you need to specify the type of eventual alignment it will be used with, rather than when the alignment is built. The HMMer profile will either be used for global or local alignment, and it will occur multiply or singly on a given sequence. All profile techniques are tremendously powerful; they can provide the most sensitive, albeit extremely computationally intensive, database similarity search possible.

Finally, we can use multiple sequence alignments to infer phylogeny. Based on the assertion of homologous positions in an alignment, many, many different methods can estimate the most reasonable evolutionary tree for that alignment. A few of the packages that incorporate these methods were mentioned earlier in the complications sections with regard to format issues: PAUP* (Swofford, 1989-2007), MrBayes (Ronquist and Huelsenbeck, 2003), and PHYLIP (Felsenstein, 1980-2007). This is a huge, complicated, and highly contentious field of study. (See the Woods Hole Marine Biological Laboratory's excellent summer short course, the *Workshop on Molecular Evolution*, at http://workshop.molecularevolution.org/.) However, always remember that regardless of the algorithm used, any form of parsimony, all of the distance methods, all maximum likelihood techniques, and even all types of Bayesian phylogenetic inference, they all make the absolute validity of your input alignment matrix their first and most critical assumption (but see Lunter, et al., 2005).

Therefore, the accuracy of your multiple sequence alignment is the most important factor in inferring reliable phylogenies; your interpretations are utterly dependent on its quality. Structural alignments are the 'gold-standard,' but the luxury of having homologous solved structures is not always available. In fact, many experts recommend not using any questionable portions of sequence data at all. These highly saturated regions have the property known as 'homoplasy.' This is a region of a sequence alignment where so many multiple substitutions have occurred at homologous sites that it is impossible to know if those sites are properly aligned, and thus, impossible to ascertain relationships based on those sites. Phylogenetic inference algorithms' primary assumption is most violated in these regions, and this phenomenon increasingly confounds evolutionary reconstruction as divergence between the members of a dataset increases. Because of this, only analyze those sequences and those portions or your alignment that assuredly do align. If any sequences or portions are in doubt, exclude them. This usually means trimming down or masking the alignment's terminal ends and may require internal trimming or masking as well. These decisions are somewhat subjective by nature, experience helps, and some software, such as ASaturA (Van de Peer, et al., 2002) and T-Coffee (Notredame, et al. 2000),

has the ability to evaluate the quality of particular regions of your alignment as well. Biocomputing is always a delicate balance — signal against noise — and sometimes it can be quite the balancing act!

The T-Coffee shop

I call this section the T-Coffee shop because T-Coffee (Notredame, et al. 2000) is much more than merely a program for doing multiple sequence alignment. Much like a Starbucks® coffee shop offers many different flavors and types of coffee drinks, the T-Coffee command line offers an entire suite of multiple sequence alignment tools. Notredame (2006) has done a very good job of providing documentation with the package's distribution. In particular be sure to read the entire *Tutorial and FAQ*. It's quite good (albeit with some English language mistakes and confusions), and I can't do it justice in my description here. Therefore, I will attempt to distil out the most vital portions of the documentation, and illustrate a subset of T-Coffee's potential in a 'bare-bones' manner, just enough to get a novice user started in exploring the package.

As mentioned in the algorithm section, T-Coffee is one of the most accurate multiple sequence alignment tools around, and it does this in its default mode. It achieves its accuracy by producing the multiple sequence alignment that has the highest consistency level with a library of preprocessed, global and local pairwise alignments. However, it can do much more than that. In addition to merely aligning a sequence dataset, it can combine preexisting alignments, evaluate the consistency of alignments, extract a series of motifs to create a local alignment, perform all sorts of data manipulation and format operations, and with SAP (Structure Alignment Program, Taylor, 1999) installed it can even use structural information to make the most accurate protein alignment possible.

T-Coffee's "seq_reformat" tool can perform standard data reformatting operations and change the appearance of your alignment, but it is incredible, as well, for extracting or combining subsets of your data based on sequence names, patterns, coordinates in the alignment, and/or level of consensus. It even has the ability to translate DNA sequences into their corresponding protein sequences, or to generate DNA alignments based on the corresponding protein alignment, either using the actual DNA sequences (*ala* mrtrans and relatives) or using a random back-translation procedure. Furthermore, "seq_reformat" can read phylogenetic trees in Newick format to compare two trees or to prune tips off of a tree. Another practical utility in T-Coffee is "extract_from_pdb;" it allows you to download either the three-dimensional coordinates or the FastA format sequence of structures held at the PDB (Protein Data Bank, http://www.rcsb.org/pdb/, Berman, et al., 2003) using UNIX's "wget" command. There is little that cannot be done with the T-Coffee utility tools when it comes to data organization and manipulation. One of my favorites is using it to remove sequences that are, or are nearly, redundant. Pages 11 through 25 of Notredame's tutorial cover all these operations very well, and I encourage you to work through the examples there; I will not take the space to review them here.

Okay, how do we begin? I'm making the presumption that you already have T-Coffee installed on either your own computer or on a server that you have access to. If this is not the case, then refer to Notredame's (2006) *Technical Documentation*, and either install it yourself, or get a local systems administrator to do it for you. I'm

using version 5.05 here. I'll be consistent in my command syntax in these examples, but realize that the Notredame's tutorial mixes up command syntax a bit, freely replacing equal signs and commas with spaces in some examples and not in others, and T-Coffee doesn't mind. Let's first look at T-Coffee's default mode. Issue the following command to see all of T-Coffee's default parameter settings:

```
Prompt% t_coffee -help
```

The list is huge; scroll back to skim over the entire thing. Notice the "–seq" parameter usage: "List of sequences in any acceptable format." T-Coffee will accept several input formats, but it works most reliably if you have your input files in FastA format. If I have a file containing unaligned sequences in FastA format named "unaligned.fa," then the following command will run T-Coffee without any options or parameter specifications:

```
Prompt% t_coffee unaligned.fa
```

This will produce a screen trace of the program's progress; an output alignment named "unaligned.aln" in Clustal Aln format; another alignment file named "unaligned.html" in HTML format for Web browsers, with columns color-coded based on reliability; and a file named "unaligned.dnd," that contains the Newick format tree used to guide the alignment. If you have your dataset spread around in more than one file, then you can use the "–seq" option followed by a comma-delimited list of input files. This option will also strip the gaps out of any input file that might already be an alignment. And if you don't like Clustal Aln format, use the "–output" option. Here I'll use it to generate a FastA format output alignment file named "unaligned1.fasta_aln" from two FastA input files (T-Coffee uses the first file's primary name to identify the output file). The only output alignment file produced this way is in FastA format:

```
Prompt% t_coffee -seq=unaligned1.fa,unaligned2.fa -output=fasta_aln
```

The T-Coffee "–output" option supports alignment formats with the following identifiers: "msf_aln" (for GCG MSF), "pir_aln" (for PIR), "fasta_aln" (for FastA), "phylip" (for PHYLIP), as well as its default "clustalw_aln" and "html". You can produce output files in more than one format by comma separating the identifiers. So, you can't get directly to NEXUS format, but PAUP* has the ability to import GCG MSF, PIR, or PHYLIP format with the ToNEXUS command. Plus, if you don't like T-Coffee's default output file naming convention, you can use the "–outfile" option to specify any name you might want.

Alignment parameters

As in nearly all sequence alignment programs, the substitution matrix and the gap penalties are very important run parameters. In most cases the T-Coffee matrix defaults, the BLOSUM62 and 50 matrices for its global and local pairwise alignment steps, respectively, will work just fine. And, in fact, T-Coffee only uses these matrices in its first pass through your dataset, when it builds its consistency library. It replaces the usual BLOSUM style matrix when building its final multiple sequence alignment with the optimal position specific scores of all the potential pairwise matings in its library. Regardless, using the optional "–matrix=blosum30mt" flag (or blosum40mt or blosum45mt depending on your data's level of divergence) is a great idea, whenever dealing with sequences that

are quite dissimilar. Furthermore, gap penalties can be messed with, if you really want to, but the default gap opening penalty of -50 and gap extension penalty of zero, changed through the "–gapopen" and "–gapext" options are, as Notredame (2006) says, only "cosmetic," changing the final alignment's appearance by changing how residues slide around in 'unalignable' regions, since all the 'alignable' regions are found from the previously built library. It's much more complicated if you want to change how the library alignments are built, and I don't suggest messing with it. If you insist, the parameters are specified through the "–method" option, and two methods build the pairwise alignment library by default (several others are available by option for special cases): a global, "slow_pair," one, and a local, "lalign_id_pair," one. To change their respective default behaviors a combination of "MATRIX" specification and "GOP" and "GEP" parameters are used. The defaults for the global library alignments are a "GOP" of -10 with a "GEP" of -1; and for the local alignment library a "GOP" of -10, with a "GEP" of -4. For instance, if I have a really lousy dataset, with barely discernible homology, and with no structural homologues at all, then perhaps using a combination of parameters such as the following would produce a more accurate and more pleasing looking multiple sequence alignment:

```
Prompt% t_coffee -seq=lousydata.fa -matrix=blosum30mt
   -gapopen=-100 -gapext=0 -output=fasta_aln,clustalw_aln,html
   -method=slow_pair@EP@GOP@-5@GEP@-1,lalign_id_pair@EP@GOP@-5@GEP@-4
```

Notice the bizarre syntax: the at sign, "@," is used as a method parameter separator, and "EP" stands for "Extra Parameter." This command would run both the global and local library builds with the BLOSUM30 matrix, would double the 'cosmetic' gap opening penalty, would cut the penalties in half for opening a gap in both the gloabal and local library alignments, and would keep all the extension penalties at their default levels. Additionally it would produce output alignments in FastA, ClustalW, and HTML formats. If you wanted to use different substitution matrices for the different methods, then you would add e.g. "MATRIX@blosum45mt" after "@EP@" and before "@GOP@ for the appropriate method. But, will this actually produce a 'better' alignment?

Quality

This brings up the heart of T-Coffee: consistency. T-Coffee's method relies on reconciling its internal pairwise alignment library as best as it can with its eventual multiple sequence alignment; the more these agree, the more consistent is the alignment and, we assume, the more accurate. This premise allows us to use T-Coffee to evaluate and compare alignments. The easiest way to see how accurate T-Coffee 'thinks' its alignment is, is to look at the "SCORE" it receives in its ClustalW or HTML format output. The higher this score value is, the more closely the alignment overall agrees with the internal pairwise alignment library. Notredame (2006) says that values above 40 are "usually pretty good." Every T-Coffee ClustalW and HTML format output alignment has this value associated with it. However, what if you don't have the right output format, or you want to see how the output from some other alignment program ranks, or you are interested in how different alignments compare to each other, or you are interested in what portions of an alignment are good and what portions are bad? T-Coffee can do all of this.

I'll discuss methods that do not rely on structure first. Structural methods will follow where I discuss T-Coffee's ability to integrate sequence and structure. T-Coffee's CORE index is the local consistency level of each position within your alignment. All T-Coffee HTML format output alignments represent this index with color-coding, plus there are specific score output file options. Position colors range across the spectrum from blue, to green, to yellow, to orange, and finally red, corresponding to an increase in consistency level from none to absolute. These colors correspond to local consistency values of 0 through 9. To test a preexisting alignment with the CORE index use the "–infile" specification for your alignment, the "–evaluate" (replaces the deprecated "–score" flag, and in default mode equivalent to "–special_mode=evaluate") option, and minimally specify HTML output format:

```
Prompt% t_coffee -infile=lousydata.fasta_aln -evaluate -output=html
```

Notice we need to use "–infile" rather than "–seq" in order to run T-Coffee in this manner. There are even ways to automatically filter unreliable columns from your alignment based on the CORE index; however, the various commands' syntax are quite complicated, and I refer you to page 45 of Notredame's (2006) tutorial.

Comparing alignments

T-Coffee has several ways to compare existing alignments of the same sequences beyond just looking at their consistency scores. The "aln_compare" module is one of the more powerful, and can tell you how different two alignments are. It needs to be launched with the "–other_pg" option, which tells T-Coffee that you want to use an external module. This option must be the first parameter on the command line after "t_coffee." "aln_compare" supports several further options that can help with visualization. Here the "aln_compare" module is used to analyze the difference between two existing alignments with the "–al1" and "–al2" options to produce an output screen trace of the first alignment where all residues with less than 50% of their pairing partners in the other alignment are represented as an "x:"

```
Prompt% t_coffee -other_pg=aln_compare -al1=trial1.fasta_aln
   -al2=trial2.fasta_aln -output_aln -output_aln_threshold 50
   -output_aln_modif x
```

The same command without the "–output_align" parameters will produce a summary statistic of the percentage of similarity between the two alignments counting the sum of all pairs of residues in those alignments. Type the command without any parameters to see all that "aln_compare" offers:

```
Prompt% t_coffee -other_pg=aln_compare
```

All of T-Coffee's built in external modules support this help syntax, versus its standard "–help" option.

Another way to compare alignments is to turn one into T-Coffee's library and leave the other an alignment. You need to use the "–aln" option to do this. This option tells T-Coffee to use the specified input alignment file to build its library. The following command will show how well the alignment "somedata1.fasta_aln" agrees with the library produced from the alignment "somedata2.fasta_aln:"

```
Prompt% t_coffee -infile=somedata1.fasta_aln -aln=somedata2.fasta_aln
```

```
-evaluate -output=html
```

The HTML alignment output will highlight those residues that are in agreement or not between the two input alignments using T-Coffee's standard CORE index coloring scheme.

Combining alignments

Again, there's a slew of ways to combine alignments with T-Coffee. One neat way is to not really worry how alignments compare and just turn them all into a T-Coffee library so that they will combine together yielding one optimal alignment that best agrees with all the input alignments. They do not even need to all have the same sequences to do this. Turn the three specified alignments into a library and produce an output alignment in Clustal Aln, FastA, and HTML format with the command below:

```
Prompt% t_coffee -aln=one.fasta_aln,two.fasta_aln,three.fasta_aln
   -output=clustal_aln,fasta_aln,html
```

And, of course, you could easily add some unaligned input sequences to the mix with the "–seq" option as well.

As discussed in this chapter under multiple sequence alignment applications, profiles are a very powerful technique for building larger and larger alignments. T-Coffee can deal with profiles in several ways, though they are not quite the same sort of profile as, for instance, Gribskov (et al., 1987) or Eddy (1998) envisioned. T-Coffee defines profiles as multiple sequence alignment matrices that will not have their gaps removed, rather than a true PSSM where residues receive higher weights in more conserved regions. Regardless, T-Coffee can take as many different profiles and sequences as you want to specify, and combine them all into one alignment (given that it is biologically correct to attempt to align them):

```
Prompt% t_coffee -profile=one.fasta_aln,two.fasta_aln,three.fasta_aln
   -seq=lousydata.fa,evenworsedata.fa -output=clustal_aln,fasta_aln,html
```

This command will feed three alignments to T-Coffee such that the sequences within them will not have their gaps removed, it will add more gaps to reconcile those three alignments, and it will add two more sequences to the resulting alignment in the most consistent manner.

And to get the most accurate profile alignment add "–profile_comparison=full," which runs the profile alignment in a slower, more exact mode "on a library that includes every possible pair of sequences between the two profiles," as opposed to the above command, which "vectorizes" the multiple sequence alignments designated as profiles (Notredame, 2006).

Combining methods

T-Coffee has a special 'Meta' mode named M-Coffee (Wallace, et al., 2006). This gives T-Coffee an incredible amount of power. M-Coffee is amazing for those situations where you just don't know what alignment tools to trust, and you don't want to have to build and test a bunch of alternatives. It automatically runs up to eight different multiple alignment programs (by default, more external methods can be added) on your data, and

combines the best parts of each, to come up with one, most consistent, consensus alignment. Your system needs to have ClustalW (Thompson, et al., 1994), POA (Lee, et al., 2002), Muscle (Edgar, 2004), ProbCons (Do, et al., 2005), MAFFT (Katoh, et al., 2005), Dialign-T (Subramanian, et al., 2005), PCMA (Pei, et al., 2003), and T-Coffee (Notredame, et al. 2000) all installed for this to work. If I want to see how M-Coffee handles that lousy data FastA format file I have, then I would issue the following command to run M-Coffee in its default mode:

```
Prompt% t_coffee -seq=lousydata.fa -special_mode=mcoffee
   -output=clustal_aln,fasta_aln,html
```

The output Clustal and HTML format files will list the alignment's overall score as a percentage of consistency between all the methods. The HTML format will additionally provide T-Coffee's usual color-coded position consistencies. Or, If you prefer some methods to others, you can select particular methods to combine with the "–method" option, with syntax like the following:

```
Prompt% t_coffee -seq=lousydata.fa -method=t_coffee_msa,mafft_msa,muscle_msa
   -output=fasta_aln,html
```

Here's some general guidelines as to which of T-Coffee's integrated external multiple sequence alignment methods are best in which situations (based on Notredame, 2006, and Edgar and Bataoglou, 2006):

clustalw_msa	neither the fastest nor the most accurate, but a reasonable 'industry-standard.'
probcons_msa	uses consistency and Bayesian inference to provide ultra-accurate, but very slow runs.
muscle_msa	very, very fast for large datasets, especially; uses weighted log-expectation scoring.
mafft_msa	in fast mode (FFT-NS-i) screaming quick on large datasets, but not incredibly accurate; in slow mode (L-INS-i) very accurate but quite slow, especially with large datasets.
pcma_msa	combines ClustalW and T-Coffee strategies.
poa_msa	very accurate local alignments using partial order graphs.
dialignt_msa	accurate local, segment-based, progressive alignment.

Pick and choose among the most appropriate methods and let M-Coffee combine the best aspects of each.

Local multiple sequence alignments

If you know the coordinates of some predefined sequence pattern in one sequence, you can use T-Coffee's mocca routine (Multiple OCCurrences Analysis, Notredame, 2001) to find all the occurrences of similar patterns in other sequences and assemble a local multiple sequence alignment of them. Mocca is a perl script that launches T-Coffee, computes a T-Coffee library from the input sequences, and then prompts you with an interactive menu to extract the homologous motifs and assemble the alignment. It is designed to find and align motifs of 30% and greater identity that are at least 30 amino acids long. The interactive menu can be confusing, so I recommend that you place the sequence with your identified motif first in the dataset, and specify the beginning and the length of your motif on the command line, rather than in the menu. I'll use mocca here to prepare a local sequence alignment of the motifs in a dataset named "motifdata.fa," where I know the first occurrence of the motif is at absolute position 35 and runs for 65 residues in my first sequence of the dataset:

```
Prompt% t_coffee -other_pg=mocca motifdata.fa -start=35 -len=65
```

You cannot use the "–seq" option with mocca to specify your input file in this command. Specifying your motif coordinates is also a bit tricky, since all the input sequences have their gaps removed (if it was an alignment) and are then concatenated together. That's why it's easiest if you put your known motif sequence first. After mocca computes the optimal local alignment with your motif it pauses and displays its menu. Type a capital "X" to exit the program and write the default Clustal Aln and HTML alignment files.

The 'gold-standard:' creating structure based alignments

As mentioned earlier, you need to minimally have SAP (Taylor, 1999) installed on your system for T-Coffee's structure based alignment mode to actually use structural information. And, even better yet, have the FUGUE package (Shi, et al., 2001) installed as well. Additionally you need "wget" on your system, to access PDB files over the Internet, but most all UNIX/Linux installations should include this utility. T-Coffee uses a special mode named 3DCoffee (O'Sullivan, et al., 2004) to create structure-based alignments. By default 3DCoffee uses four methods to create the T-Coffee consistency library, if you specify an input sequence dataset: the standard T-Coffee global "slow_pair" and local "lalign_id_pair," SAP's "sap_pair," and FUGUE's "fugue_pair." If I have a dataset that includes some PDB structures, and those sequences are named using PDB's identifier with a chain name (e.g. 1EFTA for chain A of the *Thermus aquaticus* elongation factor Tu structure, Kjeldgaard, 1993), then the following command will produce the most consistent alignment of them based on all available structural pairs and T-Coffee's usual pairs:

```
Prompt% t_coffee -seq=elongation.fa -special_mode=3dcoffee
```

You'll get three output files by default with this command: "elongation.aln," "elongation.html," and "elongation.dnd". The alignment files will have T-Coffee's standard reliability index. If you specify your input dataset is already an alignment, then the local "lalign_id_pair" will not be used, and the alignment will be turned directly into T-Coffee's library along with the SAP and FUGUE pairs:

```
Prompt% t_coffee -aln=elongation.fasta_aln -special_mode=3dcoffee
```

These 3DCoffee analyses will even produce output alignments if you don't have SAP or FUGUE installed, but they will report error warnings for every pair, and, naturally, no structural information will be used in the production of the alignment. If you don't like the warning messages, just specify the particular methods you have available, e.g:

```
Prompt% t_coffee -aln=elongation.fasta_aln -method=sap_pair,slow_pair
```

A nice trick is to combine two existing, related, but only distantly so, alignments with 3DCoffee, if they both have at least one sequence whose structure has been solved, and they follow proper naming conventions. Suppose I have one alignment of elongation factor Tu sequences containing the sequence for the solved structure for *Thermus aquaticus*, and another alignment of elongation factor 1α sequences containing the sequence of the solved human structure. Do this analysis with the "–profile" input specification:

```
Prompt% t_coffee -profile=elongation1a.fasta_aln,elongationtu.fasta_aln
   -special_mode=3dcoffee
```

The output alignment will combine the two existing alignments in the most consistent manner with the structural alignment of the human and *Thermus aquaticus* sequences.

Using structure to evaluate alignments

Your existing alignment needs to have at least two members with solved structures in order to evaluate it using T-Coffee's structural method, and, as above, those sequences need to be named according to their PDB identifiers as well as their chain. T-Coffee uses a special version of Root Mean Square Distance Deviation analysis not dependent on specific α carbon backbone superpositioning called iRMSD (the "i" stands for intra-catener, Armougom, et al., 2006) for structural alignment evaluation. It also reports a normalized NiRMSD not dependent on the alignment's length, and it reports an older measure, APDB, not as powerful as iRSMD, based on the fraction of residue pairs with 'correct' structural alignments. The smaller the iRMSD Å numbers are, the bigger the APDB percent will be, and the better the alignment corresponds to structural 'reality.' As with the other integrated external methods in the T-Coffee package, iRMSD is launched with the "–other_pg" option. Specify your alignment's file name with the "–aln" option, and specify an output file with the "–apdb_outfile" option, otherwise the output will just scroll to screen. Optionally generate an HTML alignment output as well with "–outfile":

```
Prompt% t_coffee -other_pg=irmsd -aln=elongation.fasta_aln
   -apdb_outfile=irmsd.out -outfile=irmsd
```

The summary statistics at the end of the output file are the most telling:

```
#TOTAL for the Full MSA
        TOTAL     EVALUATED:  81.58 %
        TOTAL     APDB:       78.25 %
        TOTAL     iRMSD:       0.75 Angs
        TOTAL     NiRMSD:      0.93 Angs

# EVALUATED: Fraction of Pairwise Columns Evaluated
# APDB:      Fraction of Correct Columns according to APDB
# iRMDS:     Average iRMSD over all evaluated columns
# NiRMDS:    iRMSD*MIN(L1,L2)/Number Evaluated Columns
# Main Parameter: -maximum_distance 10.00 Angstrom
# Undefined values are set to -1 and indicate LOW Alignment Quality
# Results Produced with T-COFFEE (Version_5.05)
# T-COFFEE is available from http://www.tcoffee.org
```

My elongation factor 1α/Tu alignment is pretty darn good with an overall NiRSMD of less than 1 Å and an APDB statistic of 78%. The old T-Coffee external method specification "–other_pg=apdb" produces the same output. The optional HTML output shows which residues in the solved structures (only) are in agreement with the structural alignment using the APDB coloring scheme where blue corresponds to 0% and red corresponds to 100% (get iRSMD coloring with "–color_mode=irmsd").

Special situations

Even though T-Coffee is great for so many things, its default run parameters are far from ideal for some special situations. It doesn't work very well for huge datasets, that is anything with over about 100 sequences, it doesn't particularly like DNA/RNA alignments, and it has some problems with jumping over huge gaps like you would see when aligning splicing variants or cDNA to genomic DNA containing introns. Fortunately there are ways around each one of these scenarios, and most all multiple sequence alignment programs have trouble with the same situations as well.

Let's start with real large alignments. Both Muscle and MAFFT (in fast mode) are more appropriate for datasets with more than around 100 sequences; however, T-Coffee does have a special mode that will at least allow it to estimate an approximate alignment with such datasets. This should work when your dataset has an overall identity of 40% or more:

```
Prompt% t_coffee -seq=hugedata.fa -special_mode=quickaln
```

The resulting alignment should be about as accurate as one built with ClustalW. For datasets with between 50 and 100 sequences T-Coffee automatically switches to another heuristic mode named DPA (double progressive alignment).

Naturally DNA and RNA alignments are way harder for T-Coffee to perform. The rationale for this phenomenon is explained in the introduction, and it confounds every single multiple sequence alignment program around. DNA is just really hard to align unless the sequences are 90% or so identical. One way to help T-Coffee with an alignment that must be built using DNA or RNA, because the locus does not code for proteins, is to specify that the sequences are DNA with the "–type" option:

```
Prompt% t_coffee -seq=DNAdata.fa -type=dna
```

T-Coffee should detect this sequence type automatically, but it can't hurt to declare it up front. When T-Coffee realizes that it is working with DNA it uses specific DNA optimized methods to build its library, "slow_pair4dna" and "lalign_id_pair4dna." These methods have lower built-in gap penalties and use a DNA specific scoring matrix. And if your DNA is a particularly noisy coding locus, but you just can't figure out the translation, because it is so noisy, then T-Coffee's special "cdna_fast_pair" method takes potential amino acid similarity considering frameshifts into account, and may help:

```
Prompt% t_coffee -seq=noisy_cDNAdata.fa -method=cdna_fast_pair
```

You may want to try this when aligning cDNAs to genomic DNA as well, in order to jump over the introns. If the results look good, you can even use T-Coffee's external "seq_reformat" module to translate it to the appropriate protein translation in spite of how noisy the original DNA sequences were:

```
Prompt% t_coffee -other_pg=seq_reformat -in=noisy_cDNAdata.fasta_aln
   -action=+clean_cdna,+translate > noisy_pep.fasta_aln
```

Notice that "seq_reformat" does not support "–outfile;" you need to use UNIX ">" output redirection. "+clean_cdna" is a small HMM that tries to maximize the appropriate frame choice at every point in the sequences to best match the alignment of all the other sequences, and "+translate," obviously, does the translation.

T-Coffee is actually pretty good at jumping over big gaps in protein sequence alignments, such as can be present when trying to align various protein splicing variants. It achieves this by relying heavily on the local, pairwise knowledge gained in its internal "lalgn_id_pair" method. If you can't find the alignment using default parameters, try to restrict your method to the local pair library only:

```
Prompt% t_coffee -seq=splicingvariant.fa -method=lalgn_id_pair
```

Another trick that can work well with EST sequences is to increase the default ktuple size from 2 to 5, along with specifying the "cfasta_pair_wise" "dp_mode" option:

```
Prompt% t_coffee -seq=EST.fa -dp_mode=cfasta_pair_wise -ktuple=5
```

This should use a "checked" (Notredame, 2006) version of the FastA algorithm with a word size of five to create T-Coffee's consistency library. A DNA alignment can be produced much faster this way than with other methods given sufficient similarity, but difficult regions will end up less accurate. Try mixing and matching the various methods most appropriate for your data to come up with your 'most satisfying' multiple sequence alignment.

T-Coffee servers and Expresso

Finally, if all this command line stuff just bewilders you, there are several T-Coffee Web servers around, e.g. the primary one at http://www.tcoffee.org/; they just can't do all the things that can be done in the package from the command line. T-Coffee Web servers do, however, offer "Regular" and "Advanced" modes. Furthermore, as of this writing, T-Coffee Web servers are the only way to run Expresso. Expresso is T-Coffee's latest and greatest mode. It's the triple-shot espresso, extra whip cream, Irish whiskey enriched, grandé cappuccino, of modes! It's a logical pipeline: the server runs an all against all BLAST search of your dataset against the sequences in PDB, finds all templates with greater than 60% identity to any of your sequences (if they exist), and then uses T-Coffee/3DCoffee to align your dataset using that structural information to build the consistency library as well as T-Coffee's usual library methods, all in an automated fashion. Give it a try. It's very slick, and impressively accurate.

Conclusion

The comparative method is a cornerstone of the biological sciences, and key to understanding systems biology in so many ways. Multiple sequence alignment is the comparative method on a molecular scale, and is a vital prerequisite to some of the most powerful biocomputing analyses available, such as structure/function prediction and phylogenetic inference. Understanding something about the algorithms and the program parameters of multiple sequence alignment is the only way to rationally know what is appropriate. Knowing and staying well within the limitations of any particular method will avert a lot of frustration. Furthermore, realize that program

defaults may not always be appropriate. Think about what these default values imply and adjust them accordingly, especially if the results seem inappropriate after running through a first pass with the default parameters intact. T-Coffee's and others' consistency based approaches can help with these decisions.

Oftentimes you'll need to deal with quite complicated datasets — distantly related local domains, perhaps not even in syntenic order between sequences; or widely divergent paralogous systems resulting from large gene expansions; or extremely large sequence collections with megabases of genomic data; often you'll even need to resort to manual alignment, at least in some regions — these are the situations that will present vexing alignment problems and difficult editing decisions. These are the times that you'll really have to think. A comprehensive multiple sequence editor such as GCG's SeqLab, or alternative freeware and public-domain editors, can be a lifesaver in these situations. As can the powerful evaluation and comparison modes built into the T-Coffee multiple sequence alignment package.

References

Altschul, S.F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990) Basic Local Alignment Tool. *Journal of Molecular Biology* **215**, 403–410.

Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D.J. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research* **25**, 3389–3402. Server at http://www.ncbi.nlm.nih.gov/BLAST/, and source code at ftp://ftp.ncbi.nih.gov/blast/.

Armougom, F., Moretti, S., Keduas, V., and Notredame, C. (2006) The iRMSD: a local measure of sequence alignment accuracy using structural information. *Bioinformatics* **22**, 35–39.

Bailey, T.L. and Elkan, C., (1994) Fitting a mixture model by expectation maximization to discover motifs in biopolymers, in *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*, AAAI Press, Menlo Park, California, U.S.A. pp. 28–36.

Bailey, T.L. and Gribskov, M. (1998) Combining evidence using p-values: application to sequence homology searches. *Bioinformatics* **14**, 48–4.

Bairoch A. (1992) PROSITE: A Dictionary of Sites and Patterns in Proteins. *Nucleic Acids Research* **20**, 2013–2018.

Berman, H.M., Henrick, K., and Nakamura, H. (2003) Announcing the worldwide Protein Data Bank. *Nature Structural Biology* **10**, 980.

Bininda-Emonds, O.R.P. (2005) transAlign: using amino acids to facilitate the multiple alignment of protein-coding DNA sequences. *BioMed Central Bioinformatics* **6**, 156. Available through http://www.personal.uni-jena.de/~b6biol2/ProgramsMain.html.

Clamp, M., Cuff, J., Searle, S. M. and Barton, G. J. (2004), The Jalview Java Alignment Editor, *Bioinformatics* **20**, 426–427. Available at http://www.jalview.org/.

Cole, J.R., Chai, B., Farris, R.J., Wang, Q., Kulam-Syed-Mohideen, A.S., McGarrell, D.M., Bandela, A.M., Cardenas, E., Garrity, G.M., and Tiedje, J.M. (2007) The ribosomal database project (RDP-II): Introducing myRDP space and quality controlled public data. *Nucleic Acids Research* **35**,169–172. See http://rdp.cme.msu.edu/.

Do, C.B., Mahabhashyam, M.S.P., Brudno, M., and Batzoglou, S. (2005) ProbCons: Probabilistic Consistency-based multiple sequence alignment. *Genome Research* **15**, 330-340. Available at http://probcons.stanford.edu/download.html.

Eddy, S.R. (1996) Hidden Markov models. *Current Opinion in Structural Biology* **6**, 361–365.

Eddy, S.R. (1998) Profile hidden Markov models. *Bioinformatics* **14**, 755–763.

Edgar, R.C. (2004) MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research* **32**, 1792-1797. Available through http://www.drive5.com/muscle/.

Edgar, R.C. and Batzoglou, S. (2006) Multiple sequence alignment. *Current Opinion in Structural Biology* **16**, 368–373.

Etzold, T. and Argos, P. (1993) SRS — an indexing and retrieval tool for flat file data libraries. *Computer Applications in the Biosciences* **9**, 49–57.

Felsenstein, J. (1980-2007) PHYLIP (Phylogeny Inference Package) version 3.6. Distributed by the author. Department of Genome Sciences, University of Washington, Seattle, Washington, U.S.A. Available at http://evolution.genetics.washington.edu/phylip.html.

Feng, D.F. and Doolittle, R. F. (1987) Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *Journal of Molecular Evolution* **25**, 351–360.

Galtier, N., Gouy, M. and Gautier, C. (1996) SeaView and Phylo_win, two graphic tools for sequence alignment and molecular phylogeny. *Compututer Applications in the Biosciences* **12**, 543–548. Available through http://pbil.univ-lyon1.fr/software/.

Genetics Computer Group (GCG®), (Copyright 1982-2007) *Program Manual for the Wisconsin Package®*, version 11, Accelrys, Inc., San Diego, California, U.S.A. See http://www.accelrys.com/products/gcg/.

Gilbert, D.G. (1990–2006) ReadSeq. Distributed by the author. Biology Department, Indiana University, Bloomington, Indiana, U.S.A. See http://iubio.bio.indiana.edu/soft/molbio/readseq/.

Gonnet, G.H., Cohen, M.A., and Benner, S.A. (1992) Exhaustive matching of the entire protein sequence database. *Science* **256**, 1443–1145.

Gotoh, O. (1982) An improved algorithm for matching biological sequences. *Journal of Molecular Biology* **162**, 705–708.

Gribskov, M., McLachlan, M., and Eisenberg, D. (1987) Profile analysis: detection of distantly related proteins. *Proceedings of the National Academy of Sciences U.S.A.* **84**, 4355–4358.

Gupta, S.K., Kececioglu, J.D., and Schaffer, A.A. (1995) Improving the practical space and time efficiency of the shortest-paths approach to sum-of-pairs multiple sequence alignment. *Journal of Computational Biology* **2**, 459–472. MSA available at http://www.ncbi.nlm.nih.gov/CBBresearch/Schaffer/msa.html.

Guex, N., Diemand, A., and Peitsch, M.C. (1999) Protein modelling for all. *Trends in the Biochemical Sciences* **24**,364–367. See http://swissmodel.expasy.org//SWISS-MODEL.html

Henikoff, S. and Henikoff, J.G. (1992) Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences U.S.A.* **89**, 10915–10919.

Higgins, D.G., Bleasby, A.J., and Fuchs, R. (1992) CLUSTALV: improved software for multiple sequence alignment. *Computer Applications in the Biological Sciences* **8**, 189–191.

Hofmann, K. and Baron, M. (1999) BOXSHADE server at http://www.ch.embnet.org/software/BOX_form.html; software available at ftp://www.isrec.isb-sib.ch/pub/boxshade.

Karlin, S. and Altschul, S.F. (1990) Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proceedings of the National Academy of Sciences U.S.A.* **87**, 2264–2268.

Katoh, K., Kuma, K., Toh, H., and Miyata T. (2005) MAFFT version 5: improvement in accuracy of multiple sequence alignment. *Nucleic Acids Research* **33**, 511–518. Available at http://align.bmr.kyushu-u.ac.jp/mafft/software/.

Kjeldgaard, M., Nissen, P., Thirup, S., and Nyborg, J. (1993) The crystal structure of elongation factor EF-Tu from *Thermus aquaticus* in the GTP conformation. *Structure* (London, England) **1**, 35–50.

Lee, C., Grasso, C., and Sharlow, M. (2002) Multiple sequence alignment using partial order graphs. *Bioinformatics* **18**, 452–464. POA available at http://bioinfo.mbi.ucla.edu/poa/.

Letondal, C. and Schuerer, K. Pasteur Institute, Paris, France, http://www.pasteur.fr/english.html. ProtAl2DNA available at ftp://ftp.pasteur.fr/pub/GenSoft/unix/alignment/protal2dna.

Lunter, G., Miklos, I., Drummond, A., Jensen, J.L., and Hein, J. (2005) Bayesian coestimation of phylogeny and sequence alignment. *BioMed Central Bioinformatics* **6**, 83

National Center for Biotechnology Information (NCBI) *Entrez*, public domain software distributed by the authors. http://www.ncbi.nlm.nih.gov/Entrez/ National Library of Medicine, National Institutes of Health, Bethesda, Maryland, U.S.A.

Needleman, S.B. and Wunsch, C.D. (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology* **48**, 443–453.

Notredame, C. (2001) Mocca: semi-automatic method for domain hunting. *Bioinformatics* **17**, 373–374.

Notredame, C. (2006) *T-Coffee: Tutorial and FAQ* and *Technical Documentation*. Included with the distribution through http://www.tcoffee.org/Projects_home_page/t_coffee_home_page.html.

Notredame, C., Higgins, D.G., and Heringa, J. (2000) T-Coffee: a novel method for multiple sequence alignments. *Journal of Molecular Biology* **302**, 205-217.

Olsen, G. (1992) lecture: *Inference of Molecular Phylogenies*, University of Illinois at Urbana-Champaign, U.S.A.; Thursday, September 3, 1992.

O'Sullivan, O., Suhre, K., Abergel, C., Higgins, D.G., and Notredame, C. (2004) 3DCoffee: combining protein sequences and structures within multiple sequence alignments. *Journal of Molecular Biology* **340**, 385–395.

Pearson, W.R. (1990) Rapid and sensitive sequence comparison with FASTP and FASTA. *Methods in Enzymology* **183**, 63–98.

Pearson, W.R. (1998) Empirical statistical estimates for sequence similarity searches. *Journal of Molecular Biology* **276**, 71–84. The complete FastA package is available through http://fasta.bioch.virginia.edu/fasta_www2/fasta_down.shtml.

Pearson, W.R. and Lipman, D.J. (1988) Improved tools for biological sequence analysis. *Proceedings of the National Academy of Sciences U.S.A.* **85**, 2444–2448.

Pei, J., Sadreyev, R., and Grishin, N.V. (2003) PCMA: fast and accurate multiple sequence alignment based on profile consistency. *Bioinformatics* **19**, 427–428. Available at ftp://iole.swmed.edu/pub/PCMA/.

Rambaut, A. (1996) Se-Al: Sequence Alignment Editor. Available at http://evolve.zoo.ox.ac.uk/software.html?id=seal.

Rice, P., Longden, I. and Bleasby, A. (2000) EMBOSS: The European Molecular Biology Open Software Suite *Trends in Genetics* **16**, 276–277. Available at http://emboss.sourceforge.net/.

Ronquist, F. and Huelsenbeck, J.P. (2003) MRBAYES 3: Bayesian phylogenetic inference under mixed models. *Bioinformatics* **19**,1572–1574. See http://mrbayes.csit.fsu.edu/.

Rozen, S. and Skaletsky, H. (2000) Primer3 on the WWW for general users and for biologist programmers. In: Krawetz, S. and Misener, S. (eds) *Bioinformatics Methods and Protocols: Methods in Molecular Biology*. Humana Press, Totowa, NJ, pp 365-386. Available at http://primer3.sourceforge.net/.

Saitou, N. and Nei, M. (1987) The Neighbor-Joining Method: A New Method of Constructing Phylogenetic Trees. *Molecular Biology and Evolution* **4**, 1406–1425.

Sayle, R.A. and Milner-White, E.J. (1995) RasMol: Biomolecular graphics for all. *Trends in the Biochemical Sciences* **20**,374–376. See http://www.umass.edu/microbio/rasmol/ and http://openrasmol.org/.

Schneider, T.D. and Stephens, R.M. (1990) Sequence logos: A new way to display consensus sequences. *Nucleic Acids Research* **18**, 6097-6100. See http://www-lmmb.ncifcrf.gov/~toms/sequencelogo.html.

Schwartz, R.M. and Dayhoff, M.O. (1979) Matrices for detecting distant relationships, in *Atlas of Protein Sequences and Structure* (Dayhoff, M.O. ed.) **5**, National Biomedical Research Foundation, Washington, D.C., U.S.A. pp. 353–358.

Shi, J., Blundell, T.L., and Mizuguchi, K. (2001) FUGUE: sequence-structure homology recognition using environment-specific substitution tables and structure-dependent gap penalties. *Journal of Molecular Biology* **310**, 243–257.

Smith, R.F. and Smith, T.F. (1992) Pattern-induced multi-sequence alignment (PIMA) algorithm employing secondary structure-dependent gap penalties for comparative protein modeling. *Protein Engineering* **5**, 35–41. Available at http://genamics.com/software/downloads/pima-1.40.tar.gz.

Smith, R.F., Wiese, B.A., Wojzynski, M.K., Davison, D.B., Worley, K.C. (1996) BCM Search Launcher — an integrated interface to molecular biology data base search and analysis services available on the World Wide Web. *Genome Research* **6**, 454–62. See http://searchlauncher.bcm.tmc.edu/.

Smith, S.W., Overbeek, R., Woese, C.R., Gilbert, W., and Gillevet, P.M. (1994) The Genetic Data Environment, an expandable GUI for multiple sequence analysis. *Computer Applications in the Biosciences* **10**, 671–675. The original Sun OS version is at ftp://megasun.bch.umontreal.ca/pub/gde/. See Linux and Mac OS X GDE ports at http://www.bioafrica.net/GDElinux/index.html and http://www.msu.edu/~lintone/macgde/.

Smith, T.F. and Waterman, M.S. (1981) Comparison of bio-sequences. *Advances in Applied Mathematics* **2**, 482–489.

Stajich, J.E., Block, D., Boulez, K., Brenner, S.E., Chervitz, S.A., Dagdigian, C., Fuellen, G., Gilbert, J.G., Korf, I., Lapp, H., Lehvaslaiho, H., Matsalla, C., Mungall, C.J., Osborne, B.I., Pocock, M.R., Schattner, P., Senger, M., Stein, L.D., Stupka, E., Wilkinson, M.D., and Birney, E. (2002) The Bioperl toolkit: Perl modules for the life sciences. *Genome Research* **12**, 1611–1618. Available at http://www.bioperl.org/.

Subramanian, A.R., Weyer-Menkhoff, J., Kaufmann, M., and Morgenstern, B. (2005) DIALIGN-T: An improved algorithm for segment-based multiple sequence alignment. *BioMed Central Bioinformatics* **6**, 66. Available at http://dialign-t.gobics.de/.

Suyama, M., Torrents, D. and Bork P. (2006) PAL2NAL: robust conversion of protein sequence alignments into the corresponding codon alignments. *Nucleic Acids Research* **34**, 609–612. See http://coot.embl.de/pal2nal/.

Swofford, D.L. 1989–2007. PAUP* (Phylogenetic Analysis Using Parsimony, and other methods) version 4.0+. Illinois Natural History Survey, 1994; personal copyright, 1997; Smithsonian Institution, Washington D.C., U.S.A., 1998; Florida State University, 2001–2007. Home page at http://paup.csit.fsu.edu/, distributed through Sinaeur Associates, Inc. at http://www.sinauer.com/ Sunderland, Massachusetts, U.S.A.

Taylor, W.R. (1999) Protein structure comparison using iterated double dynamic programming. *Protein Science* **8**, 654–665. SAP available for non-profit academic work at http://mathbio.nimr.mrc.ac.uk/ftp/wtaylor/sap/.

Thompson, J.D., Gibson, T.J., Plewniak, F., Jeanmougin, F. and Higgins, D.G. (1997) The ClustalX windows interface: flexible strategies for multiple sequence alignment aided by quality analysis tools. *Nucleic Acids Research* **24**, 4876–4882. Available at ftp://ftp-igbmc.u-strasbg.fr/pub/ClustalX/.

Thompson, J.D., Higgins, D.G. and Gibson, T.J. (1994) CLUSTALW: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, positions-specific gap penalties and weight matrix choice. *Nucleic Acids Research* **22**, 4673–4680. Available at http://www.ebi.ac.uk/clustalw/.

Van de Peer, Y., Frickey, T., Taylor, J.S. and Meyer, A. (2002) Dealing with saturation at the amino acid level: A case study based on anciently duplicated zebrafish genes. *Gene* **295**, 205–211. ASaturA available at http://bioinformatics.psb.ugent.be/software_details.php?id=6.

von Heijne G. (1987) *Sequence Analysis in Molecular Biology; Treasure Trove or Trivial Pursuit*. Academic Press, San Diego, California, U.S.A.

Wallace, I.M., O'Sullivan, O., Higgins, D.G., and Notredame, C. (2006) M-Coffee: combining multiple sequence alignment methods with T-Coffee. *Nucleic Acids Research* **34**, 1692–1699. Included in T-Coffee distribution.

Waterman, M.S. (1989) Sequence alignments, in *Mathematical Methods for DNA Sequences* (Waterman, M.S. ed.), CRC Press, Boca Raton, Florida, U.S.A.

Wernersson, R. and Pedersen A.G. (2003) RevTrans — Constructing alignments of coding DNA from aligned amino acid sequences. *Nucleic Acids Research* **31**, 3537–3539. Available at http://www.cbs.dtu.dk/services/RevTrans/download.php.

Wuyts, J., Perriere, G., and Van de Peer, Y. (2004) The European ribosomal RNA database. *Nucleic Acids Research* **32**,101–103. See http://www.psb.ugent.be/rRNA/.